commodore 64

Felhasználói kézikönyv

Felhasználói kézikönyv

Commodore 64 User's Guide

Forditotta: Czenki Zsuzsa

Marosvári Erika

Szerkesztette: dr.Varga András

Változatlan utánnyomás Tsz.: 84.258 Készült: 1000 példányban

Készült:
az Erdészeti és Faipari Tervező és Szervező Iroda
sokszorosító részlegében
Budapest VII., Csengery u. 11.
Felelős vezető: Bedő Tibor igazgató
Törzsszám: 84.162 Példányszám: 320 Terjedelme: 12,0/A/5 iv./
Felelős kiadó:
NOVOTRADE RT.
Készült a COLORPRINT G.M.
gondozásában

TARTALOMJEGYZÉK

	Oldal
BEVEZETÉS	VI
1. BEÁLLITÁS.....	1
Commodore 64 kicsomagolása és beállitása.....	2
Üzembeállitás.....	4
Opcionális kapcsolások.....	6
Müködtetés.....	8
Szin beállitás.....	11
2. NEKIINDULÁS	15
Billentyüzet.....	16
Vissza alaphelyzetbe.....	20
Programok betöltése és kimentése - kiirás és számolás.....	25
Elsőbbség	30
Dolgok egyesitése.....	31
3. A BASIC PROGRAMOZAS KEZDETE.....	33
A következő lépés.....	34
GOTO.....	34
Szerkesztési tanácsok.....	36
Változók.....	37
IF... THEN.....	40
FOR... NEXT ciklusok.....	42
4. HALADÓ BASIC.....	45
Bevezetés.....	46
Egyszerü animáció.....	46

KÖSZÖ' ZSOLT

Egymásba ágyazott ciklusok..... 48
INPUT..... 50
GET..... 53
Véletlen számok és más függvények kitalálós játék..... 57
Az Ön kockadobása..... 59
Véletlen grafika..... 60
CHR$ és ASC függvények..... 60

5. FEJLETT SZIN ÉS GRAFIKUS PARANCSOK..... 63
Szin és grafika..... 64
Szinek kiirása..... 64
Szinek CHR$ kódjai..... 66
PEEK és POKE utasitások..... 69
Képernyő grafika..... 71
Képernyő tár térkép..... 72
Szin tár térkép..... 74
További pattanó labdák..... 75

6. SZELLEM GRAFIKA..... 77
Bevezetés a szellem grafikába..... 78
Szellemek alkotása..... 79
További megjegyzések a szellemekhez.... 87
Bináris aritmetika..... 89

7. HANG LÉTREHOZÁSA..... 93
Hanghasználat, ha Ön nem számitógép programozó..... 94
A hangkészitő program szerkezete..... 94
Minta hang program..... 94
Zenélés a COMMODORE 64-n..... 95
Fontos hang beállitások..... 98

Egy dallam lejátszása a COMMODORE 64-n..... 105
Hanghatások létrehozása..... 106
Hanghatás minták kipróbáláshoz..... 108

8. HALADÓ ADATKEZELÉS..... 109
READ és DATA utasitások..... 110
Átlagok..... 112
Indexes változók..... 114
Egydimenziós tömbök..... 115
Átlagok revidiálása..... 115
Dimenzió..... 117
Kockadobás szimulálása tömbökkel..... 118
Kétdimenziós tömbök..... 120

FÜGGELÉKEK..... 125
Bevezetés..... 126
A: Commodore 64 tartozékai és szoftvere..... 127
B: Kazetta fejlett használata..... 130
C: Commodore 64 BASIC..... 132
D: BASIC kulcsszavak röviditései..... 154
E: Képernyõ kódok..... 156
F: ASCII és CHR$ kódok..... 159
G: Képernyõ és szin tártérkép..... 162
: Számitott matematikai függvények... 164
I: Csatlakozó kiosztások..... 165
J: Kipróbálandó programok..... 168
L: Hibaüzenetek..... 172
M: Hangjegy értékek..... 175
N: Irodalom..... 178
O: Szellem regiszter térkép..... 179
P: Hangvezérlõ beállitások..... 181

BEVEZETÉS

A COMMODORE 64 Felhasználói Kézikönyvet ugy tervezték, hogy minden szükséges információt megadjon Önnek berendezése helyes üzembe állitásához, megismertesse Önt a COMMODORE 64 müködésével, és hogy egyszerü, szórakoztató bevezetést adjon Önnek ahhoz, hogy megtanuljon saját programokat készitени.

Azok számára akik nem akarnak programozni megtanulni, a Commodore programok vagy mások által forgalmazott programok és/vagy játék modulok /cartridge/ használatához szükséges minden információt a könyv elejére tettünk. Ez azt jelenti, hogy nem kell az egész könyvet végigkeresni az elinduláshoz.

Vegyük sorra azokat az izgalmas sajátosságokat, melyek Önre várnak COMMODORE 64 gépében. Először, ha a grafikát tekintjük, Öné a legfejlettebb kép készitő grafika a mikroszámitógép iparban. Ezt SZELLEM GRAFIKÁnak /Sprite Graphics/ hivjuk, és lehetővé teszi, hogy 4 különböző szinben tervezze meg képeit, épp olyanokat mint amelyeket az "arcade" tipusu video játékoknál láthat. De ez még nem minden, a SZELLEM SZERKESZTŐ /Sprite Editor/ egyidőben 8 különböző képi szinten teszi lehetővé az animációt. Alkotásait bármerre elmozgathatja a képernyőn, sőt az egyik kép a másik előtt vagy mögött is elhaladhat.

Továbbá a COMMODORE 64 olyan beépitett zenei és hang hatással rendelkezik, amely versenyre kelhet több jól ismert zenei szintetizátorral. Gépének ez a része 3 különböző szólamot ad, mindegyiket teljes, zongorának megfelelő 9 oktáv terjedelemben. Továbbá 4 különböző hullámformát kap /fürészfog, háromszög, impulzus, zaj/, egy programozható hullámalak /felfutás, lecsengés, tartás, elengedés/ generátort, egy burkológörbe generátort, programozható felül- alul- és sáváteresztő szürőket változtatható rezonancia és hangerősség szabályozóval. Ha zenéjét hi-fi minőségben szeretné visz-

szajátszani, a COMMODORE 64 lehetővé teszi, hogy a hang kimenetet csaknem bármelyik jó minőségü erősitő rendszerhez csatlakoztassa.

Ha már a COMMODORE 64 más berendezésekhez történő csatlakoztatásánál tartunk, különféle, perifériákként ismert kellékek hozzácsatlakoztatásával az Ön rendszere kiterjeszthető aszerint, ahogy feladatai a bővitést igénylik. Ilyenek például a C2N kazettás egység, vagy akár 5 VIC 1541-es lemezmeghajtó egység, az Ön vagy más által készitett programok és/vagy játékok számára. Ha Önnek már van VIC 1540 lemezmeghajtó egysége ez átalakitható a COMMODORE 64 használatához. Hozzácsatlakoztatható továbbá egy VIC pontmátrix nyomtató, melyre másolatokat tud készitени programjairól, leveleiről, számláiról, stb. Ha nagyobb számitógépekhez és azok nagy adatbázisaihoz szeretne csatlakozni, csak dugaszoljon be egy VICMODEM illesztőt, és saját otthoni telefonvonalán keresztül fogadhatja a szakértők százainak és a különböző információs hálózatoknak a szolgáltatásait. Végül, ha Ön azok közé tartozik, akik a CP/M-ben használható széleskörü felhasználói szoftver iránt érdeklődnek, a COMMODORE 64 elláthatб egy bedugaszolható Z-80 mikroprocesszorral.

Ez a Felhasználói Kézikönyv segiteni fog Önnek számitógépes tudása kifejlesztésében, ami épp olyan fontos, mint a felhasználható hardver. Nem fog elmondani mindent, amit a számitógépekről tudni lehet, de kiadványok széles körére fog hivatkozni, melyek részletes információt adnak az adott témáról. A Commodore cég azt akarja, hogy Ön valóban élvezze uj COMMODORE 64-ét. És, hogy jól szórakozzék, emlékezzen rá: a programozás nem olyan dolog, amit egy nap alatt meg lehet tanulni. Legyen türelmes Önmagával, amint átveszi a Felhasználói Kézikönyvet. De mielőtt elkezdi, szánjon rá néhány percet, hogy kitöltse és postázza a tulajdonos bejelentő lapot, melyet gépével együtt kapott. Ez biztositani fogja, hogy COMMODORE 64 gépe bejegyzésre kerül a Commodore Központban és megfogja kapni a legfrissebb információkat gépének jövőbeli fejlesztési lehetőségeiről. Rendelkezésére áll a szórakozás egy egészen uj világa!

Megjegyzés:

Több program fejlesztés alatt áll, mialatt ez a kézi-

könyv készül. Kérjük kisérje figyelemmel a helyi Commodore kirendeltségénél, a Commodore Felhasználók Magazinjánál és Klubjában, mely napra készen tartja a felhasználói programok bőséges választékát, melyet széleskörben készitettek a Commodore 64-hez.

1. FEJEZET

BEÁLLITÁS

- Commodore 64 kicsomagolása és összekapcsolása
- Üzembeállitás
- Opcionális kapcsolások
- Müködés
- Szín beállitás

COMMODORE 64 KICSOMAGOLÁSA ÉS ÖSSZEKAPCSOLÁSA

A következő utasitások azt magyarázzák el, hogyan kapcsolja össze Commodore 64 gépét a televizió készülékével vagy monitorával, és hogyan győződjön meg arról, hogy minden helyesen müködik.
Mielőtt bármit hozzákapcsol számitógépéhez, ellenőrizze a Commodore 64 doboz tartalmát. E kézikönyvön kivül találnia kell:

1. Commodore 64 számitógépet
2. Tápegységet
3. Video kábelt

Ha az előbb emlitett tételek bármelyike hiányzik, azonnal vegye fel a kapcsolatot az eladóval a kiegészitési céljából.

Először nézzük meg a gépen található különböző csatlakozási lehetőségeket és ezek funkcióit.

OLDALLAP CSATLAKOZÓK

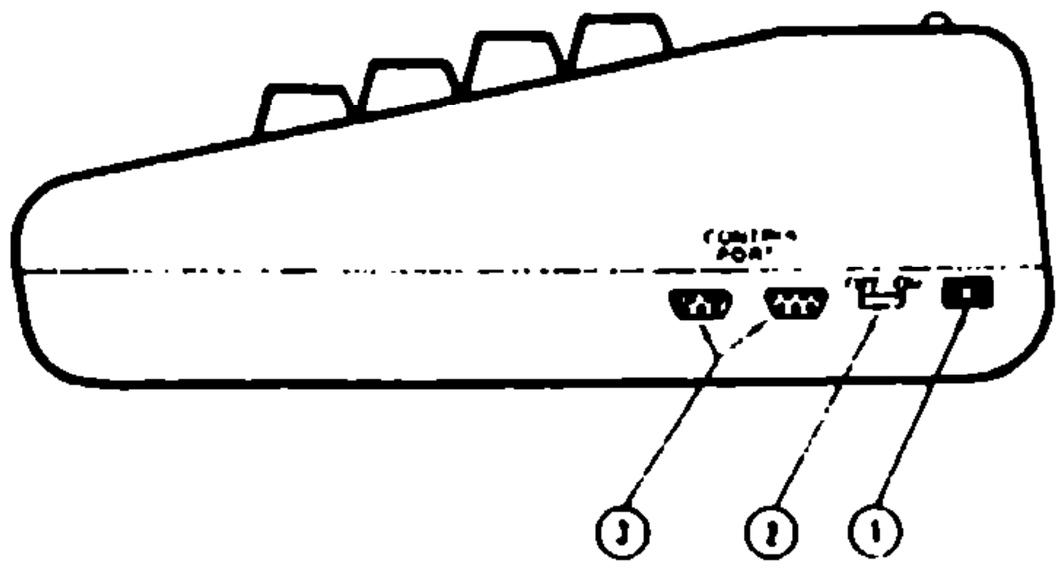
1. HÁLÓZATI CSATLAKOZÓ. A Commodore 64 energia ellátása érdekében a tápegység 9 voltos kimenetéről jövő kábel szabad végét csatlakoztatjuk ide.

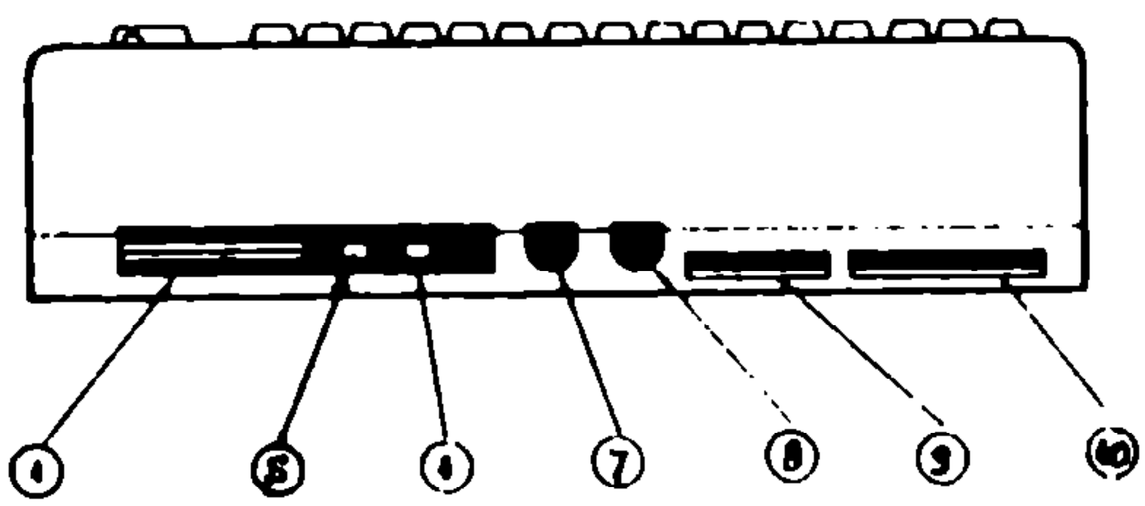
2. HÁLÓZATI KAPCSOLÓ. Bekapcsolja a Commodore 64 energia ellátását.

3. JÁTÉK CSATLAKOZÓK. Mindegyik játék csatlakozóhoz kapcsolható botkormány /joystick/, játék hajtó /paddle/ vagy fényceruza. Ezeket itt csatlakoztatjuk.

HÁTOLDALI CSATLAKOZÓK

4. MODUL CSATLAKOZÓ. A baloldalon lévő négyszögletes nyilás program vagy játék modulok csatlakoztatására ad lehetőséget.





5. CSATORNA HANGOLÓ. Ezzel az állitószervvel finoman hangolható a számitógép által a TV részére kiadott jel frekvenciája.

6. TV CSATLAKOZÓ. E csatlakozó segitségével jut el a kép és a hang a TV készülékhez.

7. AUDIO ÉS VIDEO KIMENET. Ez a csatlakozás adja a közvetlen hang jelet, melyet egy jó minőségü hangerősitő rendszerhez lehet kapcsolni, és az összetett képjelet, melyet egy televizió monitorba lehet csatlakoztatni.

8. SOROS PORT. Ezen a csatlakozón át lehet nyomtatót vagy különálló lemez egységet a Commodore 64-hez hozzákapcsolni.

9. KAZETTA INTERFÉSZ. Egy C2N kazettás egység kapcsolható itt a géphez, mely lehetőséget ad arra, hogy adatokat és programokat tároljon el szalagon /magnetofon kazettán/ későbbi felhasználás céljából.

10. FELHASZNÁLÓI PORT. Különféle felhasználói modulok csatlakoztathatók a felhasználói porton keresztül, mint pl. VICMODEM vagy RS 232 /V 24/ adatátviteli modul.

ÜZEMBEÁLLITÁS

CSATLAKOZÁS A TV KÉSZÜLÉKHEZ

A következő utasitásokat és ábrát követve csatlakoztassa a számitógépét a televizió készülékéhez vagy monitorához.

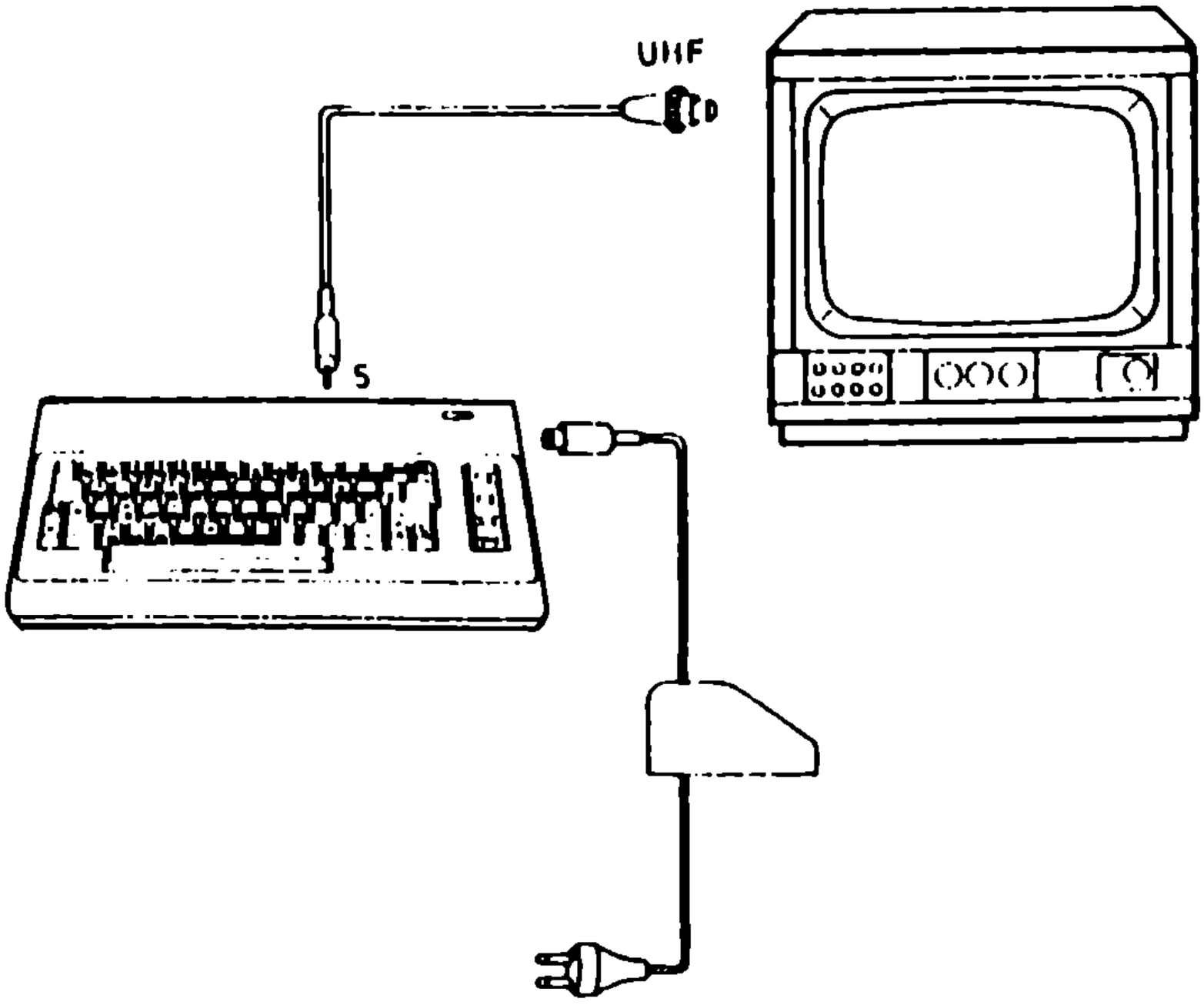
1. Csatlakoztassa a video kábel egyik végét a Commodore 64 hátoldalán lévő, az ábrán 6-os számmal jelzett, TV-jel csatlakozóhoz. A kábelnek csak az egyik vége fog illeni ide.

2. A video kábel másik végét csatlakoztassa a TV-je antenna bemenetébe. Csak be kell dugni.

3. Most dugja be a tápegységet egy megfelelő 220 V 50 Hz hálózati csatlakozóba.

4. Győződjön meg róla, hogy a számitógép kikapcsolt állapotban van és dugja be a tápegységből jövő kábelt a számitógép oldallapján lévő hálózati csatlakozóba.

5. Kapcsolja be a számitógépet. A számitógép jobb felső sarkában lévő piros lámpának ki kell gyulladnia. Ha nem igy történik, folyamodjon a fejezet végén lévő hibakereső táblához, mely felsorolja a lehetséges problémákat.



6. Kapcsolja be a televizió készülékét és válassza ki azt a csatornát, melyet nem használ televiziós műsor vételéhez. /Legtöbb TV-nek legalább négy csatorna kiválasztó gombja van./ Nagyon valószinütlen, hogy televiziója a számitógéptõl jövõ jelek vételéhez jól legyen beállitva; a hangolást most kell elvégeznie.

7. Az Ön Commodore 64-e a 36-os csatornán ad. Addig hangolja televizióját mig nem lát kék képernyõt világoskék szegéllyel és a következõ felirattal a tetején:

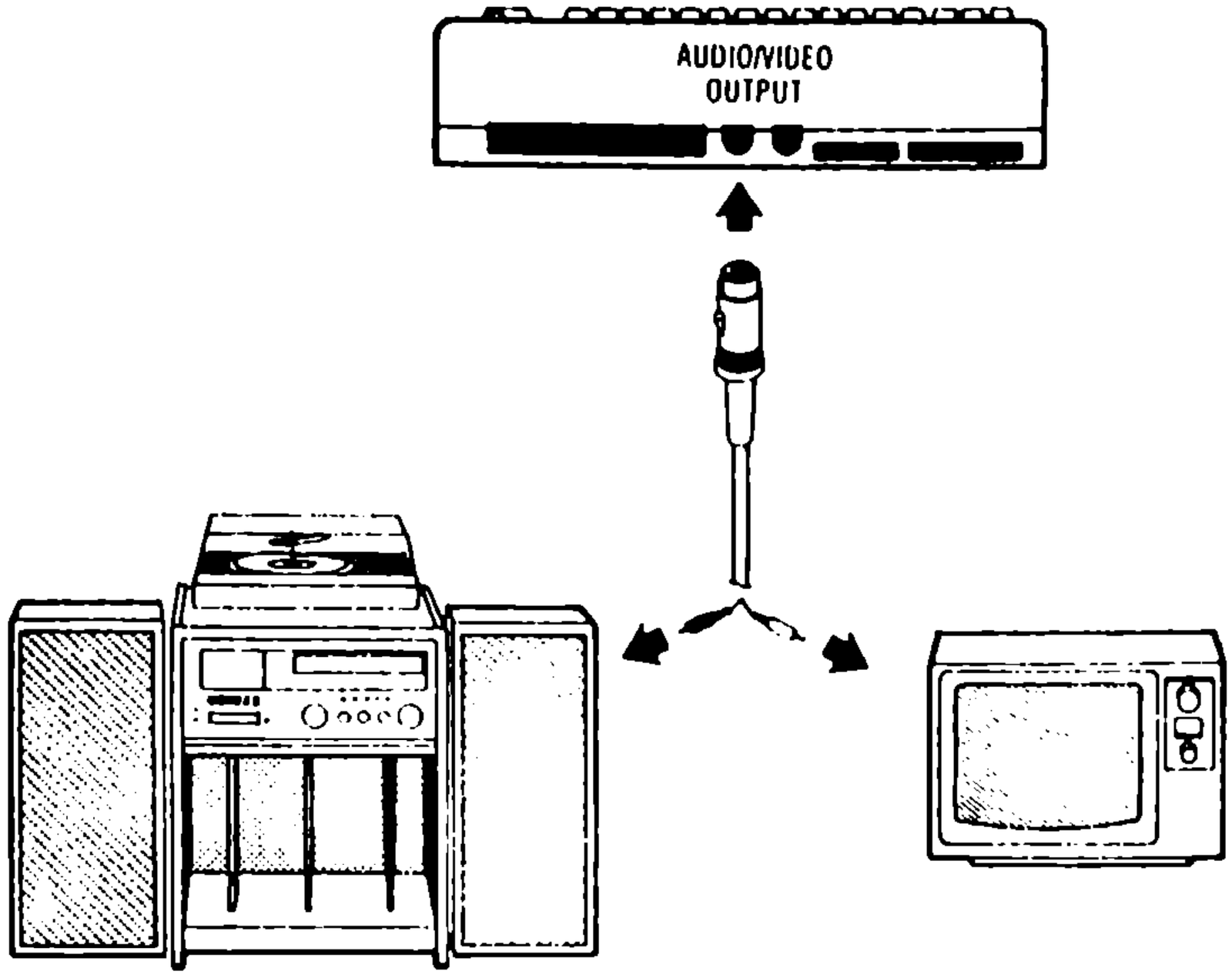
```
**** COMMODORE 64 BASIC V2 ****

64K RAM SYSTEM  38911 BASIC BYTES FREE
```

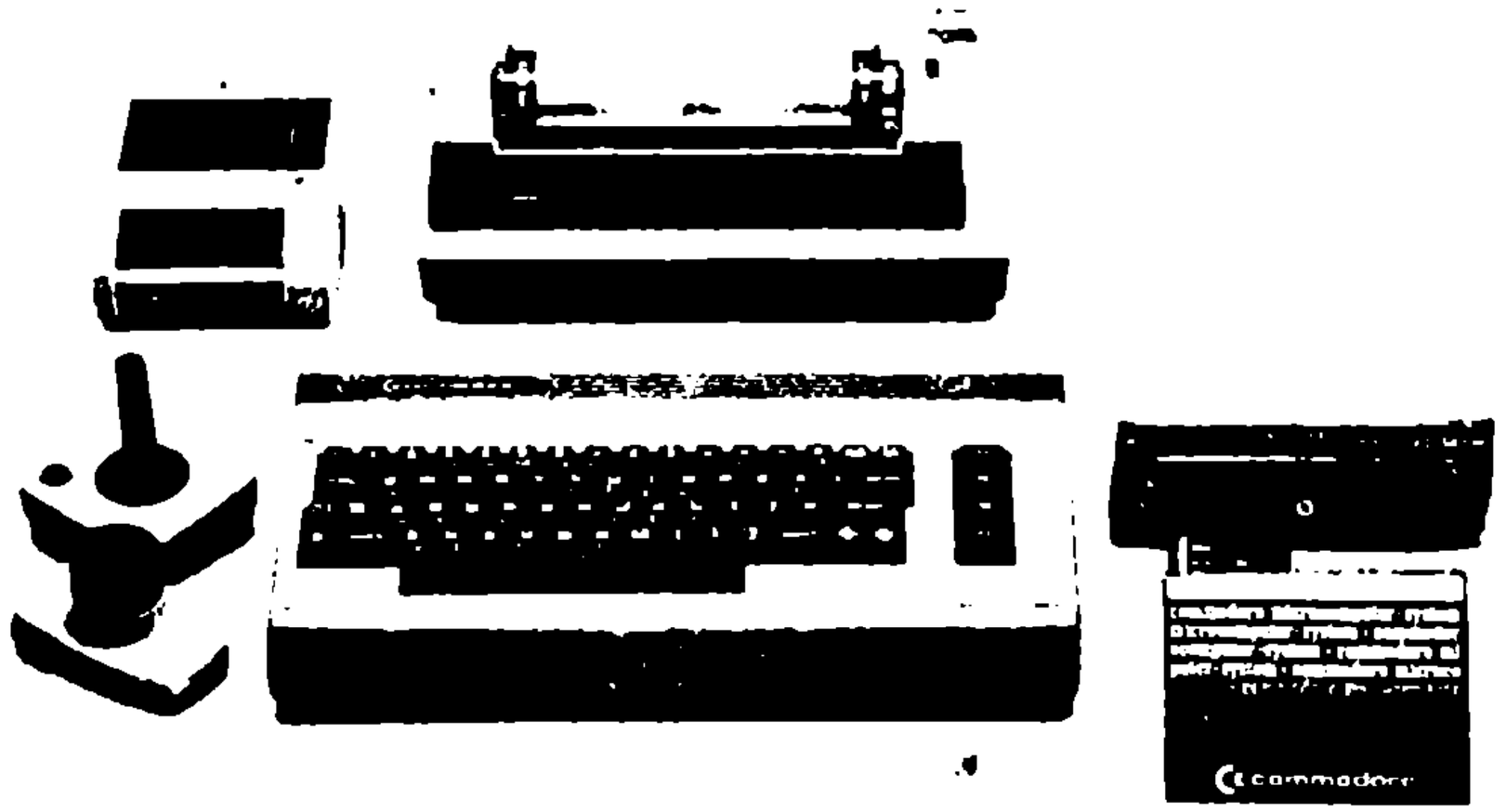
OPCIONÁLIS CSATLAKOZTATÁSOK

Mivel a Commodore 64 hi-fi minőségü csatornán szolgáltatja a hangot, a lehető legjobb minőségü hang megvalósitása érdekében Önnek lehet olyan igénye, hogy egy jó minőségü erősitőn óhajtja eljátszani azt. Továbbá, a Commodore 64 egy szabályos összetett képjelet is nyujt, mely egy televizió monitorba/vagy televizió video bemenetére/ csatlakoztatható.
Ezen opciók használatát a Commodore 64 hátoldalán lévő audio/video kimeneti csatlakozó teszi lehetővé.
E jelekhez történő hozzáférés legegyszerübb módja egy szabályos 5 polusú DIN audio kábel /nincs szállitva/ használata. Ez a kábel közvetlenül a számitógépen lévő audio/video kimenethez csatlakozik. A kábel másik végén lévő négy pólusból kettő tartalmazza az audio és video jeleket. Az 1-es függelékben lévő polus kiosztást utmutatóként használva Ön is szabadon elkészitheti saját kábelét.
Általában a DIN kábel FEKETE csatlakozója szolgáltatja a HANG jelet. Ezt a dugaszt hozzá lehet csatlakoztatni egy erősitő kiegészitő /AUXILIARY/ bemenetéhez, vagy egy monitor vagy más video rendszer - mint amilyen pl. egy képmagnó /VCR/ - AUDIO IN csatlakozójához.
A FEHÉR vagy PIROS csatlakozó rendszerint a közvetlen KÉP jelet szolgáltatja. Ez a dugasz a monitor VIDEO IN csatlakozójához vagy más video rendszerek, mint amilyen a VCR, kép bemeneti egységéhez csatlakozik.
Az Ön DIN kábelének gyártójától függően, a dugaszok szinének kódolása eltérő lehet. Ha a javasolt kapcsolást alkalmazva nem kapja meg a hang vagy kép jelet, használja az 1-es függelékben bemutatott pólus kiosztást, hogy a megfelelő pólusokat illessze.

Ha megvásárolt olyan perifériákat, minr a VIC 1541 lemez egység vagy a VIC nyomtató, lehet olyan igénye, hogy ezeket egyszerre szeretné csatlakoztatni. Ezeknek a számitógéphez történő helyes csatlakoztatása érdekében folyamodjon a kiegészitő berendezésekkel együtt szállitott felhasználói kézikönyvekhez.



Egy teljes rendszer igy nézhet ki.



A COMMODORE 64 HASZNÁLATA

1. A bal oldallapon lévő billenő kapcsolóval kapcsolja be a számitógépet.

2. Pár pillanat mulva a következő kijelzés lesz látható a TV képernyőjén:

```
**** COMMODORE 64 BASIC V2 ****

 64K RAM SYSTEM  38911 BASIC BYTES FREE

READY.
```

3. Ha TV készüléke rendelkezik kézi finom beállitó-gombbal, hangolja a TV-t amig tiszta képet nem kap.

4. A legjobb kijelzés érdekében szükséges lehet a szintelitettség utánállitására a TV-n. A későbbiekben leirt szin beállitási eljárást használhatja a helyes beállitás érdekében. Mikor az első kép megjelenik, a képernyőnek leginkább sötétkéknek kell lennie, világos kék kerettel és betükkel.

Ha nem a várt eredményt kapja, ellenőrizze a kábeleket és kapcsolásokat. A kisérő táblázat segiteni fog felderiteni bármilyen problémát.

HIBAKERESŐ TÁBLÁZAT

Tünet	Ok	Orvoslás
Jelző lámpa nem ég	Gép nincs bekapcsolva	Győződjön meg róla, hogy a hálózati kapcsoló bekapcsolt állapotban van-e.
	Hálózati kábel nincs bedugva a konnektorba	Ellenőrizze a hálózati kábelnek a hálózati dugóba való csatlakozását, nem laza-e, vagy nincs-e szétesve.
	A tápegység nincs bedugva	Ellenőrizze a kapcsolatot a fali konnektorral.
	Rossz a biztositék a számitógépben	Vigye el rendszerét biztositék cserére az erre felhatalmazott eladóhoz.
Nincs kép	A TV rossz csatornán van	Kapcsoljon át más csatornára
	Helytelen kapcsolatlétesités /a számitógép a VHF antennabemenethez csatlakozik/	A számitógépet az UHF antenna bemenethez csatlakoztassa.
	Video kábel nincs bedugva	Ellenőrizze a TV kimenet kábel csatlakozását.

Tünet	Ok	Orvoslás
A képernyõn csikos mintázat mozog	A 36. csatornán vagy közelében adó müködik	A számitógép csatornahangolójával álljon más frekvenciára, a TV finomhangolójával hangoljon utána mindaddig, mig jobb képet nem kap
Szabálytalan minta látható a TV-n modul behelyezése esetén	A modul nincs helyesen becsatlakoztatva	A feszültség kikapcsolása után helyezze be ujra a modult.
Kép szin nélkül	Rosszul behangolt TV	Hangolja ujra a TV-t.
Kép gyenge szinnel	Rossz szin beállitás a TV-n	Állitsa be a szintelitettséget, a kontrasztot és a világosságot a TV-n.
Kép tul nagy háttér zajjal	A TV hangja tulságosan fel van erõsitve	Állitsa be a TV hangerejét.
A kép rendben van, de nincs hang	A TV hangereje túl kicsi	Állitsa be a TV hangerejét.
	Kiegészitõ kimenet nincs helyesen csatlakoztatva	Csatlakoztassa a hang dugót az erõsitõ kiegészitõ bemenetéhez és kapcsolja az erõsitõt erre a bemenetet.

Tanács: A COMMODORE 64-t mindenki számára használhatónak tervezték. Azonban mi, a Commodore-nál dolgozók, ugy vettük észre, hogy a számitógép felhasználók időnként nehézségekbe ütköznek. A Commodore több kiadványt jelentetett már meg, hogy segitsen választ adni az Ön kérdéseire és hogy adjon néhány jó programozási ötletet. Az is jó ötlet lehet az Ön számára, hogy csatlakozik a Commodore Felhasználók Klubjához, hogy találkozzék más COMMODORE 64 tulajdonosokkal, akik segiteni tudnak Önnek tudása és gyakorlata gyarapitásában.

KURZOR

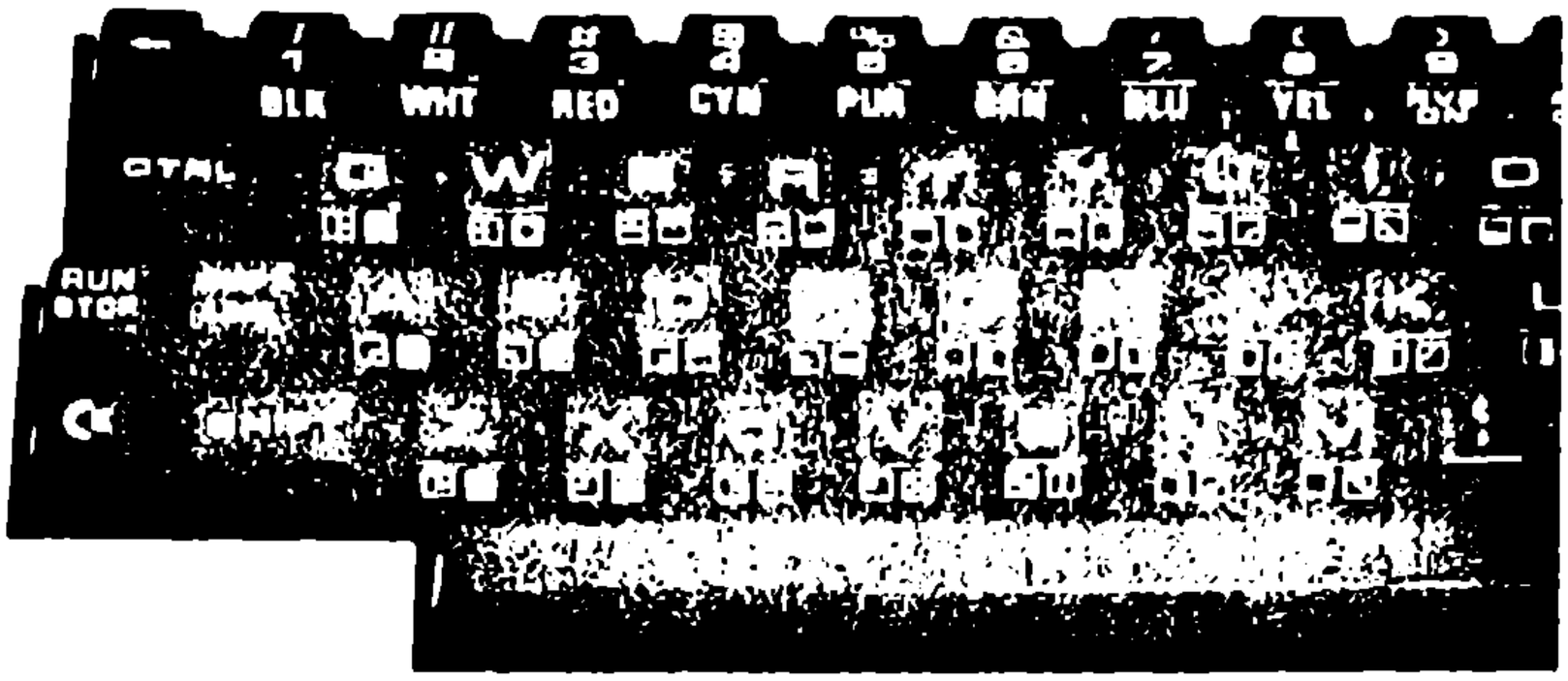
A READY /kész/ felirat után villogó négyzetet kurzornak nevezzük és azt jelzi, hogy hol fog megjelenni a képernyőn az, amit a billentyüzeten begépel. A billentyü lenyomásakor a kurzor egy helyet előre fog mozogni és az eredeti kurzor pozició az Ön által begépelt karakterrel helyettesitődik. Próbáljon meg valamit gépelni a billentyüzeten és figyelje meg, hogy a begépelt karakterek megjelennek a TV képernyőjén.

SZÍN BEÁLLITÁS

Van egy egyszerü eljárás amivel színmintát kaphat TV-jén, igy könnyen be tudja állitani készülékét. Még ha nem is otthonos azonnal a számitógép használatában, csak kövesse a menetet és meg fogja látni, milyen könnyü a Commodore 64 használata.
Először nézze végig a billentyüzet bal oldalát és állapitsa meg a *CTRL* 1/-el jelzett billentyü helyét. Ez a ConTRoL / vezérlés/ szót képviseli és más billentyükkel együttesen lenyomva arra használjuk, hogy a számitógépet különleges feladatok elvégzésére utasitsuk.

Vezérlő funkció végrehajtásához tartsa lenyomva a *CTRL* billentyüt mig lenyom egy másik billentyüt.
Próbálja ki ezt: tartsa lenyomva a *CTRL* billentyüt, mig a 9 billentyüt, aztán engedje el mindkettőt. Semmi szembeszökőnek nem kell történnie, de ha most bármely billentyüt megnyomja, a képernyőn a karakter inverz módban fog megjelenni és nem normál módban, mint a kezdő üzenet, vagy amit korábban begépelt.

1/ A billentyüket *dőlt betüvel* és aláhuzással jelöljük, a rajtuk lévő felirattal /kivéve a *SZÓKÖZ* billentyüt, melyen nincs felirás



Tartsa lenyomva a *SZÓKÖZ* billentyüt. Mi történik? Ha helyesen hajtotta végre az elõzõ eljárást, egy világoskék csikot kell látnia, amint keresztben mozog a képernyõn majd utána lefelé a következõ sorba, egészen addig amig lenyomva tartja a *SZÓKÖZ* billentyüt.

```
**** COMMODORE 64 BASIC V2 ****

 64K RAM SYSTEM  38911 BASIC BYTES FREE

READY.
```

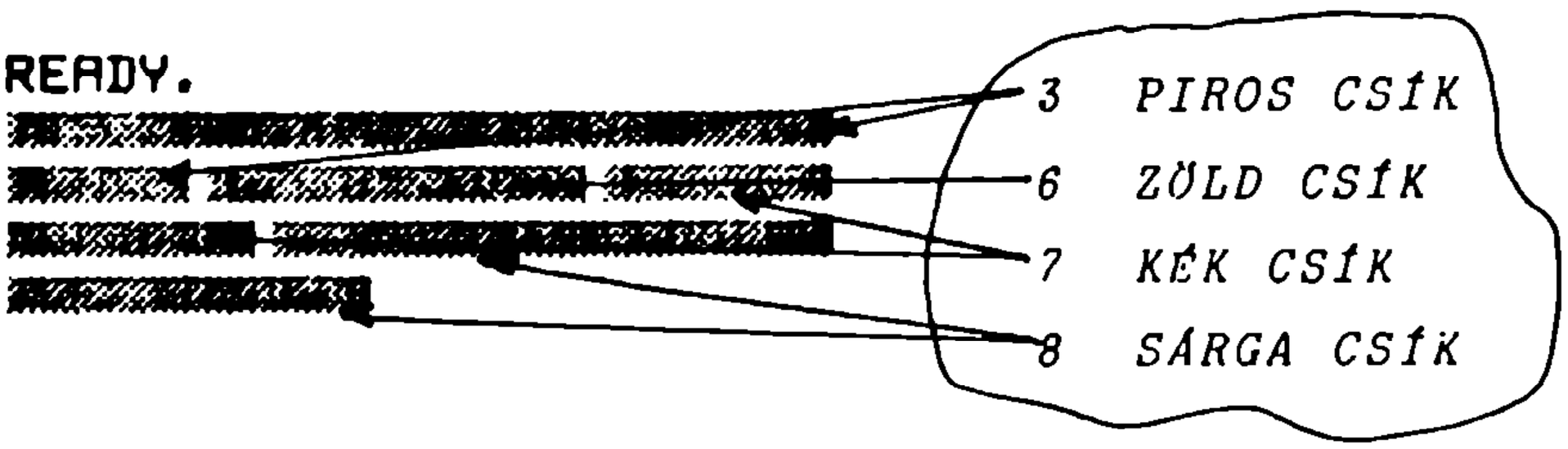
Most tartsa lenyomva a *CTRL* billentyüt mig megnyom valamilyen másik szám billentyüt. Ezek mindegyikének az elején szin jelölés látható.2/ Bármi, ami ettől kezdve megjelenik, ilyen szinü lesz. Például, tartsa lenyomva a *CTRL* és *8*-as billentyüket, majd engedje el mindkettőt. Most tartsa lenyomva a *SZÓKÖZ* billentyüt.
Figyelje a képernyőt! A csik most sárga! Hasonló módon a *CTRL* és a megfelelő billentyü lenyomásával bármely más, a szám billentyükön jelzett szinre ki tudja cserélni a csik szinét.
Változtassa a csik szinét még néhány különböző szinre, aztán állitsa be TV-jén a szinárnyalatot ugy, hogy a kijelzés megegyezzen a kiválasztott szinnel.

A kijelzés valahogy igy fog megjelenni:

```
**** COMMODORE 64 BASIC V2 ****

 64K RAM SYSTEM  38911 BASIC BYTES FREE

READY.
```



Ezen a ponton minden megfelelően van beállitva és helyesen működik. A következő fejezetek a BASIC nyelvbe vezetik be Önt. Azonban Ön közvetlenül el tudja kezdeni használni a számos Commodore 64-re előre megirt felhasználói program bármelyikét anélkül, hogy bármit is tudna a számitógép programozásról.

Ezen programcsomagok mindegyike részletes információt tartalmaz a program használatáról. Mégis azt javasoljuk, olvassa végig e kézikönyv első néhány fejezetét, hogy otthonosabbá váljon uj rendszerének alapvető működésében.

2/ Lásd a 65. oldalon lévő táblázatot

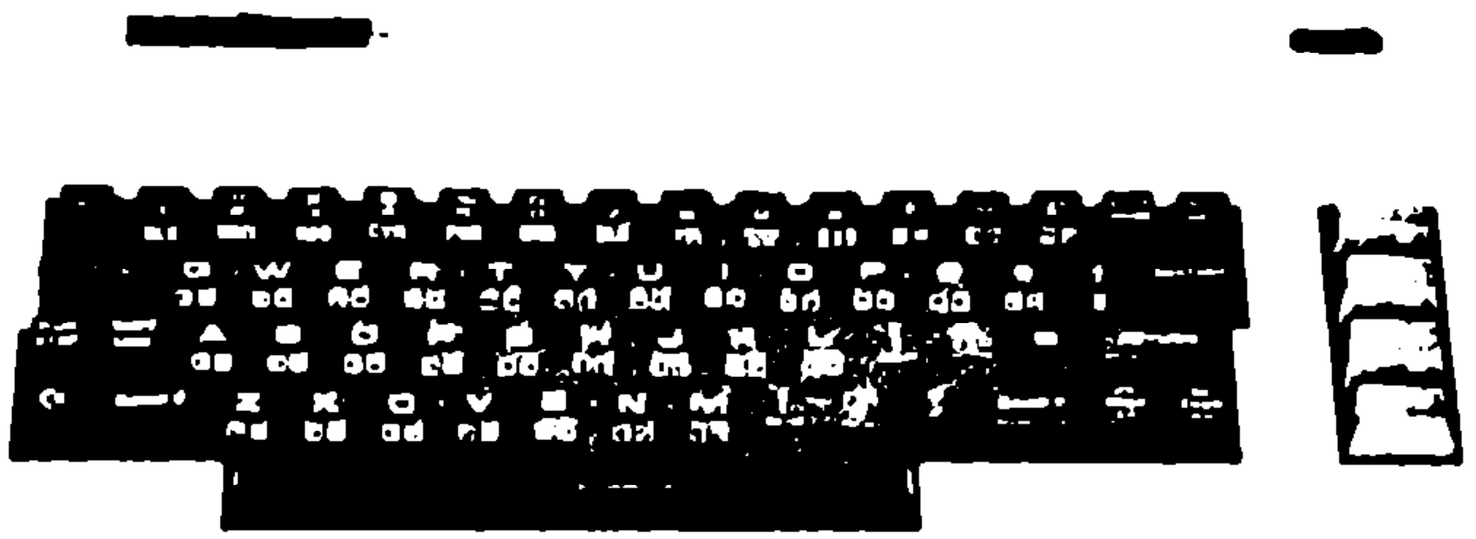
2. FEJEZET

NEKIINDULÁS

- Billentyűzet
- Vissza alaphelyzetbe
- Programok betöltése és kimentése
- Kiírás és számolás
- Elsőbbség
- Dolgok egyesitése

BILLENTYÜZET

Most, hogy mindent felállitott és beállitott, kérem szánjon egy kis időt arra, hogy hozzászoktassa magát a billentyüzethez, mivel ez jelenti az Ön számára a legfontosabb eszközt a Commodore 64-el folytatott üzenetváltásban.
Többnyire hasonlónak fogja találni a billentyüzetet a szabványos /angol/ irógép billentyüzethez. Azonban számos uj billentyü van, melyek speciális funkciókat vezérelnek. Ami most következik, az a különböző billentyüknek és feladataiknak a rövid felsorolása. A részletes müködést minden gombra vonatkozóan a következő fejezetek tartalmazzák.



RETURN /kocsi vissza-soremelés/

A *RETURN* billentyü jelzi a számitógépnek, hogy vizsgálja meg az Ön által begépelt információt és tegye be ezt az információt a tárba.

SHIFT /betüváltó/

A *SHIFT* billentyü ugyanúgy müködik, mint a szabványos irógépen. A legtöbb billentyü két betü vagy irásjel és két grafikus karakter megjelenitésére képes.
A "nagybetü/kisbetü" módban[1] a *SHIFT* billentyü a szabványos nagybetü álláshoz tartozó karaktereket adja.

1/ Lásd a 19. oldalon a *C=* Commodore billentyü fejezetet.

A "nagybetű/grafika" módban a *SHIFT* billentyű a billentyű elején lévő grafikus jelek közül a jobboldalit fogja megjelentetni.

A funkció billentyűk[1] esetében a *SHIFT* a billentyű alsó részén /elején/ jelzett funkciót adja.

SZERKESZTÉS

Senki sem tökéletes, és ezt a Commodore 64 figyelembe veszi. Számos szerkesztő billentyű ad módot a gépelési hibák kijavitására és az információk mozgatására a képernyőn.

CRSR /kurzor/

Két *CRSR* -el jelölt billentyű van /CuRSoR/, az egyik le és fel felé nyilakkal ⇓ *CRSR* ⇑, a másik balra és jobbra mutató nyilakkal ⇐ *CRSR* ⇒ . Ezek a billentyük a kurzor fel és le illetve balra és jobbra mozgatására használhatók. *SHIFT* billentyü nélkül a *CRSR* billentyük a kurzor le és jobbra mozgatását engedik meg. A *SHIFT* és *CRSR* billentyüket együtt használva, a kurzor fel vagy balra mozgatását teszik lehetővé. A kurzor billentyük különleges ismétlési képességgel rendelkeznek, miáltal a kurzort egészen addig mozgásban tartják mig el nem ereszti a billentyüt.

INST/DEL /beszurás/törlés/

Ha megnyomja az *INST/DEL* billentyüt, a kurzor egy helyet visszafelé mozog, letörölve /DELeting/ az előzőleg begépelt karaktert. Ha éppen egy sor közepén tart, a baloldali karakter törlődik, és a jobbra levők együttesen lépnek balra, hogy ne maradjon köz.
A *SHIFT* és *INST/DEL* együttes megnyomása információnak a sorba történő beszurására /INSerT/ ad lehetőséget. Például, ha gépelési hibát vett észre a sor elején - mondjuk lehagyta egy név egy részét - a ⇐ *CRSR* ⇒ billentyü használatával vissza tud menni a hibához és utána az *INST/DEL* megnyomásával be tud szurni egy betü helyet. Aztán csak be kell gépelni a hiányzó betüt![2]

1/ Lásd a 18.oldalt.

2/ Ha egy szóközt akar beszurni, ezt le is kell ütni, annak ellenére, hogy az *INST* leütése után látszólag ott szóköz keletkezett. Általánosan: ahányszor lenyomta az *INST* billentyüt, annyiszor kell a keletkezett üres helyekre valamit begépelni.

CLR/HOME /törlés/kiinduló helyzet/

A *CLR/HOME* a képernyõ kiinduló /"HOME"/ poziciójába helyezi a kurzort, mely a bal felsõ sarkot jelenti. A *SHIFT*-el együtt megnyomva a *CLR/HOME* billentyüt, le fogja törölni /CLeaR/ a képernyõt és a kiinduló pozicióba fogja helyezni a kurzort.

RESTORE /visszaállitás/

A *RESTORE* ugy müködik, mint amire a neve utal. Visszaállitja a számitógépet abba a normál állapotba, amelyben azelõtt volt, mielõtt egy programmal vagy parancsokkal Ön megváltoztatta volna a dolgokat. A késõbbi fejezetek még többet fognak errõl szólni.

FUNKCIÓ BILLENTYÜK

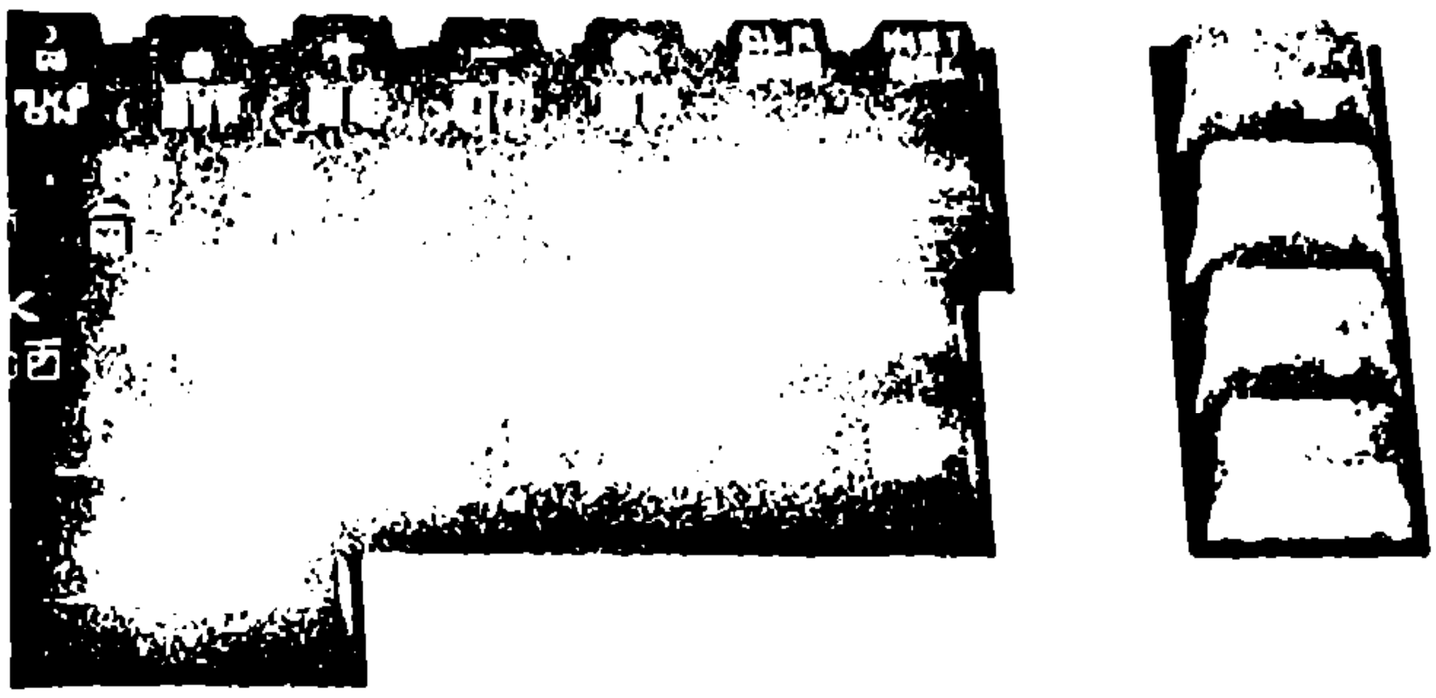
A billentyüzet jobb oldalán lévõ négy funkció billentyüt különféle tevékenységekre lehet "beprogramozni". Sokféleképpen értelmezhetõk ismétlõdõ feladatok kezelésére.

CTRL /vezérlés/

A *CTRL* billentyüt, mely a vezérlés /ConTRol/ szót képviseli, lehetõvé teszi a szin beállitást és más különleges tevékenységeket hajt végre. Vezérlõ tevékenység érdekében tartsa lenyomva a *CTRL* billentyüt mig megnyomja a kiválasztott billentyüt. A beállitási eljárás alkalmával lehetõsége volt a *CTRL* billentyü kipróbálására, amikor megváltoztatta a szöveg szinét, hogy különbözõ szinü csikokat hozzon létre.

RUN/STOP /fut/állj/

Általában a *RUN/STOP* billentyü megnyomása leállitja egy BASIC program végrehajtását. Jelzi a számitógépnek, hogy állitsa le amit csinál. A *SHIFT*-el együtt használva a *RUN/STOP* billentyü lehetõvé teszi, hogy automatikusan betöltsön egy programot szalagról.



C= COMMODORE BILLENTYŰ

A Commodore billentyűnek számos rendeltetése van. Először, lehetővé teszi a szöveges és grafikus kijelzési mód közötti cserét.

Amikor bekapcsolja a számitógépet, először nagybetü/grafikus módban van, ami azt jelenti, hogy minden amit begépel, nagybetüvel jelenik meg. Amint már emlitettük a *SHIFT* billentyü használata ebben a módban a billentyük elejének jobb oldalán lévő grafikus jelek megjelenitését eredményezi.

Ha lenyomja a C= és *SHIFT* billentyüket, a kijelzés nagybetüről kisbetüre fog megváltozni. Ha most lenyomja a C= billentyüt és bármely más grafikus jelet tartalmazó billentyüt, a billentyü elejének bal oldalán feltüntetett grafikus jel kerül kijelzésre.

Ahhoz, hogy visszakerüljön felsõ állás/grafikus módba ismét nyomja le a *C=* és *SHIFT* billentyüket.

A *C=* billentyü másik rendeltetése, hogy hozzáférést biztositson egy második, ugyancsak nyolc szöveg szinbõl álló halmazhoz.
A *C=* billentyü és bármely szám billentyü lenyomását követõen bármely szöveg, amit ezután begépel, a lenyomott billentyü által kijelölt második változatu szinü lesz. Az egyes billentyükkel nyerhetõ szöveg szineket az 5. fejezet tartalmazza.

VISSZA ALAPHELYZETBE

Most, hogy lehetõsége volt a billentyüzet áttekintésére, vizsgáljunk meg néhányat a Commodore 64 további képességeibõl.

Ha a TV készülék beállitásából adodóan még mindig a képernyõn vannak a szin csikok, nyomja meg a *SHIFT* és *CLR/HOME* billentyüket. A képernyõnek le kell törölnie és a kurzor a kiinduló helyre helyezõdik / a képernyõ bal felsõ sarkába/.

Most tartsa egyszerre lenyomva a *C=* és a *?* billentyüt. Ez visszaállitja a szöveg szinét világoskékre. Még egy lépésre van szükség a normál állapot visszaállitásához. Nyomja le a *CTRL* és a *Ø* /nulla, nem *O* / billentyüt. Ez visszaállitja a kijelzési módot normál állapotba. Ha emlékszik rá, a szin csikok elõidézéséhez a *CTRL* *9*-el inverz módra váltottunk át/ a szin csikok valójában inverz kijelzésü szóközök voltak/. Ha a szin vizsgálat közben normál szöveg módban lettünk volna, a kurzor ugyan mozgott volna, de csak üres helyeket hagyott volna.

Tanács:

Most, hogy mindent megcsinált a nehezebb módon, van egy egyszerü módja is annak, hogy a gépet normál kijelzésre állitsa vissza. Nyomja meg egyszerre:

RUN/STOP és *RESTORE*

Ez törölni fogja a képernyõt és mindent normál állapotba állit vissza. Ha a számitógépben van program,az érintetlenül marad. Ezt a sorozatot érdemes megjegyezni, kü-

lönösen ha sok programot készit.

Ha abba az állapotba óhajtja visszaállitani a gépet, mintha ki lett volna kapcsolva és ujra bekapcsolva, gépelje be: SYS 64738 és nyomja meg a *RETURN*-t. Legyen óvatos e parancs használatánál! Ez ki fog törölni minden programot és információt ami éppen a gépben van.

PROGRAMOK BETÖLTÉSE ÉS KIMENTÉSE

A Commodore 64 egyik leglényegesebb tulajdonsága a programok szalagkazettára vagy lemezre történő kimenthetősége és betölthetősége. E képessége lehetőséget ad későbbi felhasználásra irt programok elmentésére, illetve a Commodore 64 használatához előre megirt programok megvételére.
Győződjön meg róla, hogy a lemez illetve a kazettás adattároló egység helyesen van csatlakoztatva.

ELŐRE ÖSSZEÁLLITOTT PROGRAMOK BETÖLTÉSE

Azok számára akik csak az előre összeállitott, modulban kazettán vagy lemezen hozzáférhető programok iránt érdeklődnek, itt van minden amit tenniük kell:

1. MODULOK:

   Programok és játékok sorozata kapható modulban a Commodore 64-hez. Programok széleskörü választéka kinálkozik, mind üzleti, mind személyi felhasználásokhoz, és a játékok épp olyanok, mint az igazi "arcade" játékok, nem utánzatok. Ezek betöltéséhez először kapcsolja be TV készülékét. Ezután kapcsolja KI Commodore 64-t. MIELŐTT KIVESZI, VAGY BETESZI MODULJÁT KI KELL KAPCSOLNIA COMMODORE 64-ÉT, KÜLÖNBEN MEGSÉRÜL A MODUL!
   Harmadszor, dugaszolja be a Commodore 64-et. Végül gépelje be a megfelelő START billentyüt, ahogy az a minden játékhoz tartozó utasitás lapon le van irva.

2. KAZETTÁK:

   Használja C2N-es kazettás egységét és a közönsé-

ges hang kazettákat, melyek az elõre összeállitott programcsomag részeként érkeztek. Gyõzõdjön meg róla, hogy a szalag teljesen vissza van tekercselve az első oldal elejére. Ezután gépelje be, hogy LOAD /tölts/. A számitógép azt fogja válaszolni, hogy PRESS PLAY ON TAPE /nyomd le a lejátszó gombot/, melyre válaszként a kazettás egység lejátszó gombját nyomja meg. Ezen a ponton a számitógép képernyõje üressé válik addig, mig a programot meg nem találja. A számitógép azt fogja kiirni, hogy FOUND programnev /megtaláltam/. Most nyomja meg a C= billentyüt. Ez fogja a programot ténylegesen betölteni a számitóbépbe. Ha le akarja állitani a betöltést, egyszerüen csak nyomja meg a *RUN/STOP* billentyüt.

3. LEMEZ:

Az elõre megirt programokat tartalmazó lemezt ugy helyezze be óvatosan a lemezmeghajtó egységbe, hogy a lemezen lévõ cimke felfelé nézzen és Ön felé essen. Keressen meg a lemezen egy kis bevágást/elõfordulhat, hogy egy kis darab szalaggal be van fedve/. Ha helyesen teszi be a lemezt a bevágás a baloldalon lesz. Ha lemezt behelyezte, a fogantyu lenyomásával zárja be az ajtaját.
Most gépelje be LOAD "program nev", 8 és nyomja meg a *RETURN* billentyüt. A lemez forogni kezd és zümmögõ hangot ad, és a képernyõn ez lesz látható /feltéve, hogy a program neve PROGRAMNEV volt/:

```
SEARCHING FOR PROGRAMNEV
LOADING

READY
```

Mikor a READY kijelzés megjelenik és a kurzor villog, gépelje be, hogy RUN, és nyomja meg a *RETURN* billentyüt, az elõre elkészitett szoftver használatra kész.

PROGRAMOK BETÖLTÉSE SZALAGRÓL

Program betöltése szalagról és lemezrõl egyaránt egyszerü. Szalag esetén tekercselje vissza a szalagot az ele-

jére és gépelje be:

```
LOAD"PROGRAMNEV"
```

Ha nem emlékszik a program nevére, csak a LOAD-t gépelje be, ekkor a szalagon lévő első program lesz betöltve a tárba.

A *RETURN* megnyomása után a számitógép ezt fogja válaszolni:

```
PRESS PLAY ON TAPE
```

A lejátszó gomb megnyomása után a képernyő üressé és a kerettel azonos szinüvé válik, mialatt a számitógép a programot keresi.

Mikor a programot megtalálta, a képernyõn meg fog jelenni:

```
FOUND PROGRAMNEV
```

A program tényleges betöltéséhez nyomja meg a *C=* billentyüt. A betöltési müvelet félbenhagyásához nyomja meg a *RUN/STOP*-t. A Commodore billentyü megnyomása esetén, a képernyő ismét a kerettel azonos szintüre változik amig a program betöltése tart. A betöltési müvelet befejeztével a képernyő ismét normál állapotba kerül vissza és a READY felirat jelenik meg ismét.

A következő program eléréséhez nyomjon meg valamilyen, a *C=* -tól különböző billentyüt és utána a *RUN/STOP*-t.

PROGRAMOK BETÖLTÉSE LEMEZRÕL

Program lemezről történő betöltéséhez ugyanezt a formát kövesse:

```
LOAD"PROGRAMNEV",8
```

A *RETURN* billentyü megnyomása után a diszk forogni, zümmögni kezd, és a képernyő a következőt mutatja:

```
SEARCHING FOR PROGRAMNEV
LOHDING

READY
■
```

A katalógus /directory/ betöltéséhez gépelje be, hogy LOAD "$", 8. Ezután gépelje be, hogy LIST és nyomja meg a *RETURN* billentyüt.

Megjegyzés:

Mikor uj programot tölt be a számitógépbe, minden előzőleg a számitógépben lévő utasitás törlődik. Uj program betöltése előtt győződjön meg róla, hogy kimentette azt, amelyiken éppen dolgozott. Ha egy programot egyszer betöltött, el tudja inditani a RUN-al, ki tudja listázni a LIST-el, vagy változtatásokat tud végrehajtani rajta és ujra ki tudja menteni az uj változatot.

PROGRAMOK KIMENTÉSE SZALAGRA

Egy program beirása után, ha óhajtja el tudja menteni szalagra. Gépelje be:

```
SAVE"PROGRAMNEV"
```

a "programnév" legfeljebb 16 karakterből állhat. A *RETURN* billentyü megnyomása után a számitógép válasza:

```
PRESS RECORD & PLAY ON TAPE
```

Nyomja meg mind a felvétel mind a lejátszó gombot a kazettás egységen. A képernyő a keret szinére váltva üressé válik.

A programnak a szalagra történő kimentésének befejeztével, ismét a READY felirat fog megjelenni jelezve, hogy elkezdhet más programmal foglalkozni, vagy éppen kikapcsolhatja a számitógépet egy időre.

PROGRAMOK KIMENTÉSE LEMEZRE

Egy program kimentése lemezre még egyszerübb. Gépelje be:

```
SAVE"PROGRAMNEV",8
```

A 8-as a lemeznek a kódja, igy tudatja a számitógéppel, hogy lemezre akarja kimenteni a programot.
A *RETURN* megnyomása után a lemez elkezd forogni és a számitógép válasza:

```
SAVING PROGRAMNEV
READY.
■
```

KIIRÁS ÉS SZÁMOLÁS

Most, hogy tuljutottunk számos nehezebb müveleten, melyekre azért van szükség, hogy az Önnek tetsző programokat meg tudja tartani, készitsünk néhány kimentésre érdemes programot.
Próbálja meg begépelni a következőt pontosan, ahogy itt áll:

```
PRINT"COMMODORE 64"
COMMODORE 64

READY.
■
```

GÉPELJE BE EZT A SORT ÉS NYOMJA LE A RETURN-T

A SZÁMITÓGÉP IRTA KI

Ha gépelési hibát vét, használja az *INST/DEL* billentyüt a kurzortól balra lévő karakter azonnali kitörlésére. Annyi karaktert tud kitörölni, amennyi csak szükséges.
Nézzük mi történt a fenti példában! Először arra utasitotta a számitógépet, hogy irja ki /PRINT/ mindazt, ami az idézőjelek között volt. A *RETURN* meg-

nyomásával megmondta a számitógépnek, hogy tegye azt, amire utasitotta, és a COMMODORE 64 kiiródott a képernyõre.

Ha a PRINT utasitást ebben a formában használja, mindaz ami az idézõjelek között van megjelenik a képernyõn pontosan ugy, ahogy begépelte.

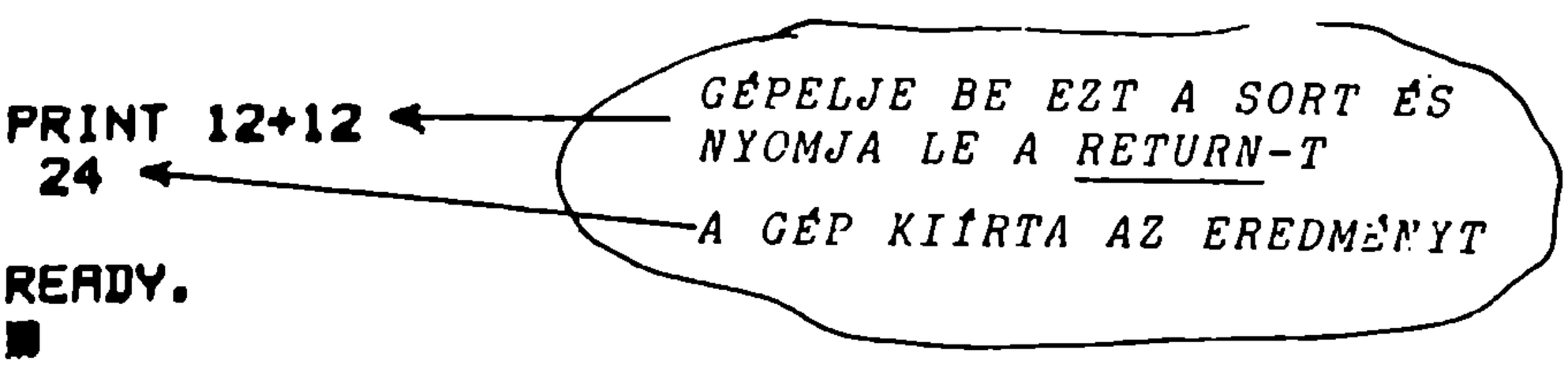
Ha a számitógép azt válaszolta, hogy:

```
?SYNTAX ERROR
```

nézze meg nem követett-e el gépelési hibát, vagy nem felejtette-e el az idézõjeleket. A számitógép preciz, és az utasitások megadását az elõirt formában várja.

De ne essen kétségbe; csak arra gondoljon, hogy ugy irja be a dolgokat, ahogy azok a példákban szerepelnek és nagyszerüen el fog boldogulni a Commodore 64-el.

Emlékezzen rá, hogy nem árthat a számitógépnek azzal, hogy begépel dolgokat. A BASIC tanulás legjobb módja, hogy megpróbál dolgokat begépelni és megnézni mi történik.
A PRINT egyike a leghasznosabb és leghathatósabb parancsoknak a BASIC nyelvben. Ezzel meg tud jeleniteni szinte mindent amit csak akar, beleértve a grafikát és a programok eredményeit.
Például próbálja ki a következõt! Törölje le a képernyõt a *SHIFT* és a *CLR/HOME* billentyük lenyomásával, és gépelje be / győzõdjön meg róla, hogy az *"1"*-es billentyüt használja az egyesnek és nem az *"I"* betüt/:



Megfigyelhette, hogy ebben az egyszerü formában a Commodore 64 egy számológép. Automatikusan kiszámolta és kiirta eredményként a "24"-et. Sõt hasonlóan elvégezhet kivonást, szorzást, osztást, hatványozást és bo-

nyolultabb matematikai függvényeket, mint például négyzetgyök számitást, stb. Ès Ön nincs korlátozva egyetlen számolásra egy sorban, de erről többet majd késõbb. Jegyezze meg, hogy a fenti formában a PRINT másképpen viselkedett, mint az elsõ példában. Ebben az esetben egy érték vagy egy számolás eredménye kerül kiirásra, nem pontosan a begépelt üzenet, mivel elhagytuk az idézõjelet.

ÖSSZEADÁS

A plusz jel /+/ jelzi az összeadást: utasitjuk a számitógépet, hogy irja ki az eredményét a 12 meg 12-nek. A többi aritmetikai müveletnek az összeadáshoz hasonló a formája. Ne felejtse el mindig megnyomni a *RETURN*-t a PRINT és a számolás begépelése után.

KIVONÁS

A kivonáshoz használja a hagyományos minusz /-/ jelet. Gépelje be:

```
PRINT 12-9 <---------- NYOMJA LE A RETURN-T
 3
```

SZORZÁS

Ha a 12-szer 12-t akarja összeszorozni, a szorzás jelölésére a csillagot /*/ használja. Begépelheti:

```
PRINT 12*12 <---------- NYOMJA LE A RETURN-T
 144
```

OSZTÁS

Az osztáshoz az ismerõs "/" jelet használja. Például, a 144-nek 12-vel történõ osztásához gépelje be:

```
PRINT 144/12 <---------- NYOMJA LE A RETURN-T
 12
```

HATVÁNYOZÁS

Hasonló módon, könnyen tud egy számot hatványra emelni / ez ugyanazt jelenti, mint egy számot önmagával szorozni egy előirt számúszor/. A "↑" /felfelé nyil/ jelzi a hatványozást.

```
PRINT 12↑5
 248832
```

Ez ugyanaz, mint begépelni azt, hogy:

```
PRINT 12*12*12*12*12
 248832
```

Tanács:

A BASIC-nek számos röviditése van a dolgok elvégzésére. Egyik ilyen mód a BASIC parancsok /vagy kulcsszavak/ röviditése. Például a ? jelet használhatjuk a PRINT helyett. Ahogy haladunk előre, több parancsnál meg fogjuk mutatni; a D függelék mutatja mindegyik parancs röviditését és azt, hogy mi fog megjelenni a képernyőn, ha begépeli a röviditett formát.

Az utolsó példa egy másik fontos sajátosságot hoz elő: ugyanabban a sorban több számitás is elvégezhető és ezek kevert tipusuak is lehetnek.
Ki tudja számolni ezt a feladatot:

```
?3+5-7+2
 3
```

EZ A ? A PRINT SZÓT HELYETTESÍTI

Egészen eddig csak kis számokat és egyszerü példákat használtunk. Azonban a Commodore 64 összetettebb számitásokra is képes.
Például össze tud adni sok számjegyü számokat. Próbálja ki ezt, de tizedes vessző helyett pontot használjon, és ne csoportositsa a sokszámjegyü szám számjegyeit /pl. a szokásos módon hármasával, ponttal elválasztva/, mert hibajelzést fog kapni.

```
7 123.45+345.78+7895.687
 8364.917
```

Ez jónak látszik, de most próbálja ki ezt:

```
7 12123123.45+345.78+7895.687
 12131364.9
```

Ha szán rá egy kis időt és összeadja ezeket kézzel, eltérő eredményt fog kapni.
Mi történik itt? Bár a számitógép több jó képességgel rendelkezik, van egy határ azokra a számokra, amelyeket kezelni képes. A Commodore 64 10 számjegyü számokkal tud dolgozni. Azonban amikor kiiratunk egy számot, csak kilenc számjegy jelenik meg.
Igy előző példánkban, az eredmény "kerekitve" lett, hogy megfeleljen a határoknak. A Commodore felfelé kerekit, ha a következő számjegy öt vagy ötnél nagyobb; lefelé kerekit ha a következő számjegy négy vagy ennél kisebb.

A 0.01 és 999,999,999 közé eső számok a szabványos jelölés szerint kerülnek kiirásra. E határokon kivül eső számok tudományos jelölést használva iródnak ki. A tudományos jelölés a nagyon nagy vagy kis számoknak 10 hatványaként történő kifejezésére szolgáló eljárás.

Ha begépeli:

```
7 1230000000000000000
 1.23E+17
```

Ez ugyanaz, mint a 1.23x10 ↑ 17, és a dolgok tömör felirására használjuk.

De a tudományos jelölés mellett is van határa a számitógép által kezelni képes számoknak. Ezek a határok:

Legnagyobb: ± 1.70141183E+38
Legkisebb: ± 2.93873588E-39

ELSŐBBSÉG

Ha megpróbált néhány, a korábban mutatott példáktól eltérő kevert számolást elvégezni, előfordulhatott, hogy nem a várt eredményt kapta. Az oka az, hogy a számitógép a számitási müveleteket bizonyos sorrendben hajtja végre.

Ennél a számolásnál:

20 + 8 / 2

nem tudja megmondani, hogy a válasz 24 vagy 14, amig nem tudja milyen sorrendben hajtsa végre a müveleteket. Ha a 20-hoz hozzáadja a 8-at 2-vel osztva /azaz 4-et/, az eredmény 24 lesz. De, ha hozzáad 20-hoz 8-at és utána osztja el 2-vel a válasz 14. Próbálja ki a példát és nézze meg milyen eredményt kap.
24-et kapott eredménynek, mert a Commodore 64 a müveleteket balról jobbra tekintve a következők szerint hajtja végre:

Először: - minusz jel, mely negativ számokat jelöl
Másodszor: ↑ hatványozás jel, balról jobbra
Harmadszor: * / szorzás és osztás, balról jobbra
Negyedszer: + - összeadás és kivonás, balról jobbra.

Kövesse végig az elsőbbség szabályai szerint, és meg fogja látni, hogy a fenti példában az osztás került először végrehajtásra és utána az összeadás, igy kaptuk eredményül a 24-et.
Állitson össze néhány feladatot saját maga, és vizsgálja meg, hogy végig tudja-e követni, és a fent leirt szabályok szerint meg tudja-e jósolni az eredményt.
Az elsőbbség megváltoztatásának van egy egyszerü módja, mégpedig a zárójelek használata, annak kijelölésére, hogy melyik müveletet akarja először végrehajtani.
Például, ha el akarja osztani a 35-öt 5 plusz 2-vel, gépelje be:

```
? 35/5+2
 9
```

válaszként 35 osztva 2-vel és hozzáadva 2-t fog kapni, ami egyáltalán nem az amit akart. Ahhoz, hogy azt kapja amit valójában akart, próbálja meg ezt:

```
? 35/(5+2)
 5
```

Ebben az esetben a számitógép azt értékeli ki elõször, amit a zárójel tartalmaz. Ha a zárójelen belül zárójel van, a legbelsõ zárójelet értékeli ki elõször.

Ahol több zárójel van egy sorban, mint itt:

```
? (12+9)*(6+1)
 147
```

a számitógép balról jobbra értékeli ki azokat. Itt 21-t kell megszorozni 7-el, hogy az eredmény 147 legyen.

DOLGOK EGYESITÉSE

Még ha sok idõt is forditottunk itt olyan területekre, melyek nem tünnek tul fontosaknak, az itt leirt részletek mélyebb értelmet kapnak majd, ha egyszer elkezd programozni és akkor fogják bizonyitani felbecsülhetetlen értéküket.
Hogy néhány ötletet adjunk ahhoz, hogy hogyan illesszen dolgokat a helyükre, tekintse a következõt: hogyan tudná egyesiteni a kétféle kiiró utasitást, amit eddig vizsgáltunk, hogy sokkal értelmesebben irjon ki valamit a képernyõre?

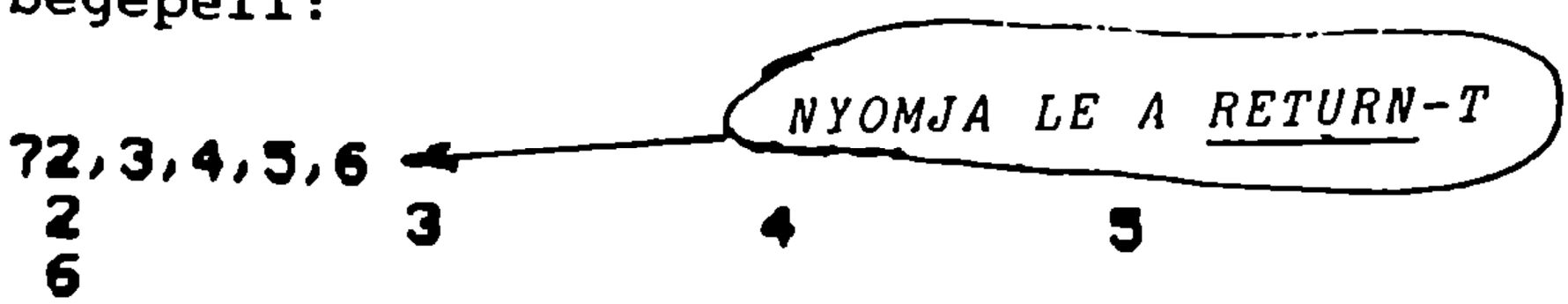
Tudjuk, hogy ha valamit idézõjelek közé zárva iratunk ki ez az információ pontosan ugy jelenik meg a képernyõn, ahogy azt beirtuk, és hogy matematikai müveletek használatával számolásokat végezhetünk. Miért ne egyesitsük a kétféle PRINT utasitást igy:

```
? "5*9=";5*9
5*9= 45
```

Még ha ez feleslegesnek is látszik egy kicsit, nem tettünk mást mint egyszerüen együtt használtuk a két fajta

kiiró utasitást. Az első rész kiirja "5%9="-et pontosan ugy ahogy begépeltük. A második rész végzi el a tényleges munkát és irja ki az eredményt. A pontosvessző választja el az utasitás üzenet részét a tényleges számolástól.
Egy kevert kiiró utasitás egyes részeit mindig el kell választani valamilyen irásjellel, hogy helyesen müködjön. Próbálja ki a vesszőt a pontosvessző helyén és nézze meg, hogy mi történik.
Az alaposság kedvéért a pontosvesszőnek az a hatása, hogy az utasitás következő része közvetlenül az előző rész után kerül kiirásra, üres helyek nélkül. A vesszső valami mást eredményez. Bár ez is egy elválasztó, ez jobban széthuzza a dolgokat.

Ha begépeli:



a számok a képernyőn keresztben kerülnek kiirásra és utána a következő sorba.
A Commodore 64 kijelzője 4 egyenként 10 oszlopos területre tagozódik. A vessző mindegyik eredményt a következő felhasználható területre tabulálja. Mivel azt kértük, hogy több információt irjon ki, mint amennyi egy sorba kifér, /öt 10 oszlopos területet próbáltunk illeszteni/ az utolsó elem lekerült a következő sorba.

A **PRINT** utasitásoknál a formázásban a vessző és pontosvessző között fellépő alapvető különbséget összetettebb kijelzések készitésénél használhatjuk fel előnyösen: ez lehetővé teszi számunkra, hogy nagyon könnyen hozzunk létre tetszetős eredményeket.

3. FEJEZET

A BASIC PROGRAMOZÁS KEZDETE

- A következő lépés
- GOTO
- Szerkesztési tanácsok
- Változók
- IF ... THEN
- FOR ... NEXT ciklusok

A KÖVETKEZŐ LÉPÉS

Egész eddig egyedül álló utasitások begépelésével egyszerü müveleteket hajtottunk végre. A *RETURN* megnyomása után az előirt müvelet azonnal végrehajtásra került. Ezt KÖZVETLEN vagy PARANCS módnak nevezzük.
De valami jelentősebb dolog megvalósitásához nem elég egy egy sorból álló utasitással müködő számitógép. Számos utasitás egyesitését programnak hivjuk, és ez lehetőséget ad a Commodore 64 összes képességeinek kihasználására.
Hogy lássa milyen egyszerü az első Commodore 64 program megirása, próbálja ki ezt:

Törölje le a képernyőt, ugy, hogy lenyomva tartja a *SHIFT* billentyüt és megnyomja a *CLR/HOME* billentyüt. Gépelje be, hogy NEW /uj/ és nyomja meg a *RETURN*-t. /Ez kitöröl minden számot, ami bennmaradhatott a számitógépben a kisérletezéseiből./
Most gépelje be a következőt pontosan ahogy itt áll /Ne felejtse el megnyomni a *RETURN*-t minden sor után/

```
10 ?"COMMODORE 64"
20 GOTO 10
█
```

Gépelje be, hogy RUN és nyomja meg a *RETURN*-t figyelje meg, mi történik. Képernyője a COMMODORE 64 felirattól fog nyüzsögni. Ha befejezte a kijelzés figyelését, nyomja meg a *RUN/STOP*-t, hogy leállitsa a programot.

Számos programozás alapjául szolgáló fontos fogalmat vezettünk be ezzel a rövid kis programmal.
Jegyezze meg, hogy minden utasitást egy szám előzött meg. Ez a SORSZÁM mondja meg a számitógépnek, hogy milyen sorrendben hajtsa végre az utasitásokat. Ezek a számok hivatkozási pontok is, amikor a programnak egy bizonyos sorra vissza kell térni. Sorszám bármilyen egész szám lehet 0-63999 között.

```
10  PRINT "COMMODORE 64
↑    └utasitás
└──── sorszám
```

```
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64

BREAK IN 10
READY.
```

Jó programozási gyakorlat a sorokat 10-esével számozni, mert később szükség lehet néhány utasitás közbeszurására.
Programunk a PRINT mellett egy másik BASIC parancsot is használt, a GOTO-t /menj/. Ez arra utasitja a számitógépet, hogy menjen közvetlenül egy bizonyos sorra és hajtsa végre azt, azután attól a ponttól folytassa.

```
10 PRINT "COMMODORE 64"
20 GOTO 10
```

Példánkban a program a 10-es sorban kiirja az üzenetet, megy a következő /20/ sorra, amelyik arra utasitja,hogy menjen vissza a 10-es sorba és ujra irja ki az üzenetet. Aztán a ciklus ismétlődik. Mivel nem adtunk módot a számitógépnek, hogy kijöjjön a ciklusból, a program vég nélkül fog körbe járni, amig fizikailag le nem állitjuk a *RUN/STOP* billentyü megnyomásával.
Ha leállitotta programját, gépelje be, hogy LIST.
Programja meg fog jelenni, érintetlenül, mivel még mindig a számitógép tárjában van. Figyelje meg azt is, hogy a számitógép átalakitotta a ? jelet PRINT-re.
Most a programot meg lehet változtatni, el lehet menteni vagy ujra végre lehet hajtani.
Másik lényeges különbség aközött, hogy valamit közvetlen módban gépelünk be vagy program formájában az, hogy ha egyszer végrehajtott és utána letörölte a képernyőt,

az utasitás elveszett. Azonban egy programot mindig vissza tud hozni a LIST begépelésével.

Megjegyzés:

Ha rövidítéseket használ ne felejtse el, hogy a számitógépnek elfogyhat a helye a sorban, ha tul sok ilyet használ.

SZERKESZTÉSI TANÁCSOK

Ha valami hibát vét egy sorban, számos javitási lehetőség áll rendelkezésre.

1. Bármikor ujra gépelhet egy sort, és a számitógép automatikusan az ujjal helyettesiti a régit /az azonos sorszámut/.

2. Egy nem kivánatos sort egyszerüen a sorszám és a *RETURN* begépelésével lehet törölni.

3. Egy létező sort is könnyen tud javitani a kurzor és szerkesztő billentyükkel.

Tegyük fel, hogy gépelési hibát vétett a példánk egyik sorában. Ahhoz, hogy az egész sor ujragépelése nélkül kijavitsa, próbálja meg ezt:

Gépelje be, hogy LIST és a *SHIFT* és ⇑ *CRSR* ⇓ billentyük együttes használatával mozgassa a kurzort amig a változtatást igénylő sorba nem ér.
Most használja a kurzor jobbra billentyüt a kurzornak a javitandó karakterhez történő mozgatásához, és gépelje rá a változást a régi karakterre. Nyomja meg a *RETURN-t* és a helyes sor a régi helyére fog kerülni. Ha több helyre van szüksége a sorban, helyezze a kurzort oda, ahol a helyre szükség van, és nyomja meg a *SHIFT* és *INST/DEL* billentyüt egyszerre, egy szóköz válik hozzáférhetővé. Ezután csak gépelje be a szükséges információt és nyomja meg a *RETURN*-t. Hasonlóan, a kurzornak a nem kivánt karakter jobb oldalára helyezésével és az *INST/DEL* billentyü megnyomásával tud karaktereket törölni.

A változtatások lezajlásának ellenőrzéséhez, gépelje be

ujra, hogy LIST, és a javitott program fog megjelenni. A sorokat nem szükséges sorszám szerinti sorrendben begépelni. A számitógép automatikusan helyes sorrendbe rakja õket.
Próbálja meg a minta programunkat kijavitani úgy, hogy megváltoztatja a 10-es sort a végére téve egy pontosvesszõt, ahogy alább megmutatjuk. Ne felejtse el a 20-as sor alá vinni a kurzort mielõtt futtatja a programot. Ezután ismét hajtsa végre a programot.

```
10 PRINT "COMMODORE  64";
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE
BREAK IN 10
READY.
```

VÁLTOZÓK

A programozási nyelvek egyik leginkább használt jellegzetessége a változók, mivel ezek sokkal több információt képviselhetnek a számitógépben. A változók mûködésének megértése a számolást egyszerübbé teszi és lehetõvé teszi olyan tettek megvalósitását, amire másképpen nem lenne lehetõség.
Képzeljünk el olyan dobozokat a számitógépben, melyek mindegyike egy számot vagy egy szöveg jeleinek karakterének füzérét /string/ képes tárolni. Ezen dobozok mindegyikét egy általunk választott névvel jelöljük. Ezt a nevet hivjuk változónak és a megfelelõ dobozban lévõ információt képviseli.

Például, ha azt mondjuk:

```
10 X%  =  15

20 X   =  23.5

30 X$  =  "X% ES X OSSZEGE"
```

A számitógép valahogy igy fogja képviselni a változókat:

```
X%   15
X    23.5
X$   X% ES X OSSZEGE
```

A változó neve képviseli a dobozt, vagy tár helyet, ahova a változó aktuális értéke eltárolódik.
Amint láthatja, egy változóhoz hozzárendelhetünk akár egy egészszámot, lebegõpontos számot,[1/] vagy egy szöveg füzért.

A változónevet követõ % jel azt jelzi, hogy a változó egész számot fog képviselni. A következõk érvényes egész tipusu változónevek:

```
A%

X%

A1%

NM%
```

A változónevet követõ $ jel azt jelzi, hogy a változó szöveg füzért fog képviselni. A következõk füzér változókra mutatnak példát:

```
A$

X$

MI$
```

1/Azért lebegõpontos, mert - amint korábban láttuk - az alkalmazható tartományon belül a számoknak mindig a 10 legértékesebb számjegyével számol a gép.Ezen belül a tizedespont, ha van, tetszõleges helyen lehet.

A lebegőpontos változók ugyanezt a formátumot követik a tipus jelzésénél:

A1

X

Y

MI

Van néhány dolog amit figyelembe kell venni a változók elnevezésénél. Elsőzör, a változónév egy vagy két karakter lehet. Az első karakternek betünek kell lennie, A és Z között; a második karakter akár betü akár számjegy /O és 9 között/ lehet. Egy harmadik karakter is hozzávehető a változó tipusának jelölésére /egész vagy szöveg füzér/, % vagy $.

Használhat kettőnél több alfabetikus karaktert tartalmazó változónevet is, de a számitógép csak az első kettőt azonositja. Igy a PO és a POSTA ugyanaz, és ugyanarra a változó dobozra vonatkozik.
A változónevekre vonatkozó utolsó szabály egyszerü: nem tartalmazhatnak semmilyen BASIC kulcsszót, mint amilyen a GOTO, RUN, stb. Folyamodjon a D függelékhez, a BASIC számára lefoglalt szavak teljes listájának megismerése érdekében.
Hogy lássa, hogyan lehet a változókkal dolgozni, gépelje be a korábban mutatott teljes programot és hajtsa végre. Ne felejtse el megnyomni a *RETURN*-t a programban minden sor után.

```
NEW

READY.
10 X%=15
20 X=23.5
30 X$="X% ES X OSSZEGE ="
40 PRINT "X%="; X%, "X="; X
50 PRINT X$; X%+X
```

Ha mindent ugy csinált, ahogy itt állt, a következő eredményt kell a képernyőre kiirva kapnia.

```
RUN
X%= 15     X= 23.5
X% ES X OSSZEGE = 38.5

READY.
■
```

Amint látja, a képernyő formázásával kapcsolatban eddig tanult minden fogást belevettünk és kiirtuk a két változó összegét.
A 10-es és 20-as sorban az X%-hoz egy egész értéket, az X-hez egy lebegőpontos értéket rendeltünk. Ez a változóhoz hozzárendelt számot beleteszi a dobozába.
A 30-as sorban egy szöveg füzért rendeltünk az X$-hoz. A 40-es sorban egy üzenet, valamint X% és X aktuális értékének kiirásához egyesitettük a kétféle PRINT utasitást. Az 50-es sor kiirja az X$-hoz rendelt szöveg füzért és X% és X összegét.
Jegyezze meg, hogy bár az X-t mindhárom változónévben használtuk, a % és $ azonositók egyedivé tették az X%, X és X$-t, igy három különálló változót jelölnek.
De a változók sokkal hasznosabbak. Ha megváltoztatja értéküket, az uj érték az eredeti érték helyére kerül ugyanabba a dobozba. Ez lehetővé teszi ilyen utasitás felirását:

```
X = X + 1
```

Ez sose lenne elfogadható a rendes algebrában, de az egyik legtöbbet használt fogalom a programozásban. Azt jelenti, hogy: vedd az X aktuális értékét, és addj hozzá egyet és helyezd az uj összeget az X-t képviselő dobozba.

IF ... THEN /ha... akkor/

A változók értékének könnyed változtatásával felfegyverkezve, kipróbálhatunk egy programot, mint ez:

```
NEW

READY.
10 SZ=0
20 ?"COMMODORE 64"
30 SZ=SZ+1
40 IF SZ<5 THEN 20
50 END
RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64

READY.
■
```

Ezzel bevezettünk két uj BASIC utasitást, és elláttuk némi vezérléssel a fejezet elején bevezetett kis, meg nem álló kiiró programunkat.

Az IF ... THEN némi logikát tesz a programba. Azt éri el, hogy HA /IF/ a feltétel igaz, AKKOR /THEN/ csinál valamit. Ha a feltétel már nem igaz, AKKOR a program következő sorát végzi el.
Számos feltétel felállitható az IF ... THEN utasitás használatával:

JELEK	JELENTÉS
<	Kisebb mint
>	Nagyobb mint
=	Egyenlő
< >	Nem egyenlő
> =	Nagyobb vagy egyenlő
< =	Kisebb vagy egyenlő

Ezek közül bármely feltétel használata egyszerü, mégis meglepően hasznos.

```
   10 SZ=0
┌→ 20 ?"COMMODORE 64"
│  30 SZ=SZ+1
└← 40 IF SZ<5 THEN 20
     |
     ↓
   50 END
```

A minta programban egy "ciklus"-t készitettünk, ami némi kényszert alkalmaz rajta, elmondva: HA egy érték kisebb valamilyen számnál AKKOR csináljon valamit.

A 10-es sor az SZ /számláló/-t beállitja 0-val egyenlőre. A 20-as sor kiirja az üzenetünket. A 30-as sor hozzáad egyet az SZ változóhoz. Ez a sor számolja, hogy hányszor hajtottuk végre a ciklust. Minden alkalommal, mikor a ciklust végrehajtja, az SZ eggyel nő.

A 40-es sor az ellenőrző sorunk. Ha az SZ kisebb mint 5, ami azt jelenti, hogy kevesebb mint 5-ször hajtottuk vég végre a ciklust, a program visszamegy a 20-as sorba és ujra kiir. Amikor az SZ egyenlő lesz 5-tel — jelezve, hogy 5 COMMODORE 64 kiirás megtörtént — a program az 50-es sorba megy, mely a program végét jelzi.

Próbálja ki a porgramot és nézze meg, hogyan gondoltuk. A 40-es sorban az SZ határértékének a kicserélésével tetszőleges számu sor irható ki.

Az IF ... THEN-nek sok más felhasználása létezik, melyeket későbbi példákban fogunk bemutatni.

FOR ... NEXT CIKLUSOK

A FOR ... NEXT ciklus használatával egyszerübben, és sokkal inkább előnyben részesitett módon véghezvihető, amit az előző példában tettünk. Tekintse a következőt:

```
NEW

READY.
10 FOR SZ = 1 TO 5
20 PRINT"COMMODORE 64"
30 NEXT SZ
RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Amint látja, a program sokkal kisebb és közvetlenebb lett. Az SZ 1 értékkel indul a 10-es sorban.[1] Aztán a 20-as csinál egy kiirást. A 30-as sor megnöveli az SZ-t 1-el. A NEXT /következő/ utasitás a 30-as sorban automatikusan visszaküldi a programot a 10-es sorba, ahol a FOR ... NEXT utasitás FOR része van. Ez az eljárás addig fog folytatódni, mig az SZ el nem éri a TO-val megadott határt.

A FOR ... NEXT ciklusban használt változó 1-nél kisebb vagy nagyobb értékkel is növelhető, ha szükséges.

Próbálja ezt:

```
NEW

READY.
10 FOR SZ = 1 TO 10 STEP 0.5
20 PRINT SZ,
30 NEXT SZ
RUN
 1            1.5          2            2.5
 3            3.5          4            4.5
 5            5.5          6            6.5
 7            7.5          8            8.5
 9            9.5          10
```

Ha begépeli és végrehajtja ezt a programot, 1-től 10-ig 0.5 lépésközzel fogja látni a számokat kiirva a képernyőre.[2]

Amit itt csinálunk semmi más, mint az SZ értékeinek kiiratása amelyeket felvesz a ciklus müködése folyamán. Azt is előirhatja, hogy a változó növekedjen vagy csökkenjen. Helyettesitse a következővel a 10-es sort:

```
10  FOR SZ = 10 to 1 STEP -.5
```
[3]

és figyelje meg, hogy az ellenkezője történik, mivel az SZ 10-től 1-ig megy csökkenő 0,5 lépésközzel.

1/ FOR SZ = 1 TO 5 jelentése: SZ=1-től 5-ig tedd
2/ STEP jelentése lépés
3/ A tizedes pont elé nem kötelező kiirni a 0-t

4. FEJEZET

HALADÓ BASIC

- Bevezetés
- Egszerű animáció
- Egymásba ágyazott ciklusok
- INPUT
- GET
- Véletlen számok és más függvények
- Kitalálós játék
- Az Ön kockadobása
- Véletlen grafika
- CHR$ és ASC függvények

BEVEZETÉS

A következő néhány fejezet azok számára iródott, akik aránylag otthonossá váltak a BASIC nyelvben és haladóbb programok irásához szükségük van a fogalmakra.

Azok, akik csak épp most kezdték el tanulni a programozást, az információk egy részét kicsit tul szakmainak találhatják a megértéshez. De bátorság ... mert a két szórakoztató fejezet, a SZELLEM GRAFIKA és a HANG GENERÁLÁS, bemutat néhány egyszerü példát, melyek az uj felhasználóknak készültek. A példák jó ötletet fognak adni arra, hogy hogyan használja gépén a kifinomult hang és grafikai lehetõségeket.

Ha ugy határoz, hogy többet akar a BASIC programirásról megtanulni, egy irodalomjegyzéket /N függelék/ tettünk a könyv végére.

Ha már otthonos a BASIC programozásban, ezek a fejezetek segitségére lesznek a haladó BASIC programozási technika elsajátitásában. Több részletes információt a COMMODORE 64 PROGRAMOZÓI KÉZIKÖNYVBEN talál.

EGYSZERÚ ANIMÁCIÓ

Gyakoroljuk a Commodore 64 néhány grafikai lehetõségét, hozzávéve néhány uj fogalmat az eddig tanultakhoz. Ha szorgalmas, gépelje be a következõ programot és nézze meg mi történik.

Észre fogja venni, hogy a kiiró utasitásokba kurzor vezérlõ és képernyõ parancsokat is beletehetünk. Ezek a PRINT utasitást követõ idézõjelek között inverz karakterként vagy inverz grafikus jelként jelennek meg. Ha ilyeneket lát a program listában, a következõ táblázat szerint járjon el:

"■"=SHIFT+CLR/HOME=CLR(TORLES)
"■"=←CRSR→=CRSR→(KURZOR JOBBRA)
"■"=SHIFT+←CRSR→=←CRSR(KURZOR BALRA)
"■"=↓CRSR↑=↓CRSR(KURZOR LE)
"■"=SHIFT+↓CRSR↑=CRSR↑(KURZOR FEL)

Tartsa pl. lenyomva a *SHIFT* billentyüt és nyomja meg a ← *CRSR* → billentyüt. A képernyő a "kurzor balra" grafikai megfelelőjét /egy inverz függőleges csikot/ fogja mutatni. Ugyanilyen módon, a *SHIFT* és a *CLR/HOME* megnyomására egy inverz szivet fog látni.

```
NEW

READY.
10 REM VISSZAPATTANO LABDA
20 PRINT"■" :REM "■ = CLR"
25 FOR X = 1 TO 10 : PRINT"■";: NEXT:
   REM "■ = CRSR LE"
28 REM A LABDA MOZGATASA BALROL JOBBRA
30 FOR LB = 1 TO 39
40 PRINT"■■";:REM "■ = SHIFT+Q,
   ■ = CRSR BAL"
50 FOR ID =1 TO 5
60 NEXT ID
70 NEXT LB
75 REM A LABDA MOZGATASA JOBBROL BALRA
80 FOR LB = 1 TO 39
90 PRINT"■■■";
100 FOR ID = 1TO 5
110 NEXT ID
120 NEXT LB
130 GOTO20
```

A : UJ UTASITÁST JELEZ

E SZÓKÖZÖK SZÜKSÉGESEK

A program egy pattanó labdát fog megjelentetni, amint balról jobbra, és vissza mozog keresztben a képernyőn.
Ha figyelmesen megnézi a /következő ábrán látható/ programot láthatja, hogy hogyan valósitja meg ezt a mutatványt.
A 10-es sor egy megjegyzés /REMark/ ami csak azt mond-

ja el, hogy mit csinál a program; ez nincs hatással magára a programra. A 20-as sor letöröl minden információt a képernyőről.

```
10 REM VISSZAPATTANO LABDA
20 PRINT"■":REM "■ = CLR"
25 FOR X = 1 TO 10 : PRINT"■";: NEXT:
   REM "■ = CRSR LE"
28 REM A LABDA MOZGATASA BALROL JOBBRA
30 FOR LB = 1 TO 39
40 PRINT" ■■";:REM "■ = SHIFT+Q,
   ■■ = CRSR BAL"
50 FOR ID =1 TO 5
60 NEXT ID
70 NEXT LB
75 REM A LABDA MOZGATASA JOBBROL BALRA
80 FOR LB = 1 TO 39
90 PRINT" ■■■■";
100 FOR ID = 1TO 5
110 NEXT ID
120 NEXT LB
130 GOTO20
```

A 25-ös sor 10 kurzor le parancsot hajt végre. Ez épp a képernyő közepére helyezi a labdát. Ha a 25-ös sort kihagynánk, a labda a képernyő legfelső sorában mozogna. A 30-as sorban egy ciklust hoztunk létre a labda 40 oszlopnyival történő balról jobbra mozgatásához.

A 40-es sor sok mindent csinál. Először egy szóközt ir ki, amellyel letöröli az előző labda helyeket, aztán kiirja a labdát, és végül egy kurzor balra müveletet hajt végre, hogy minden rendben legyen a labda aktuális helyén való ujbóli törléséhez.

Az 50-es 60-as sorban lévő ciklus egy kicsit lelassitja a labdát vagyis késlelteti a programot. Enélkül a labda tul gyorsan mozogna ahhoz, hogy lássuk. A 70-es sor befejezi azt a ciklust, amelyet a 30-as sorban kezdtünk a labda képernyőre történő kiirásához. Valahányszor csak végrehajtódik a ciklus, a labda egy hellyel jobbra mozog. Amint észrevehette a magyarázatból, egy ciklust hoztunk létre egy ciklus on belül.

Ez tökéletesen elfogadható. Az egyetlen eset amikor Ön bajba kerül az, amikor a ciklusok keresztezik egymást. A programok irásánál hasznos ellenőriznie önmagát, mint itt mutattuk, hogy meggyőződjön a ciklus logikájának helyességéről.

Hogy lássa mi történik, ha keresztezi a ciklusokat, cserélje fel a 60-as és 70-es sorban lévő utasitásokat. Hibajelzést fog kapni, mivel a számitógép zavarba jön, és nem tudja elképzelni, mi folyik itt.

A 80-tól 120-ig lévő sorok épp megforditják a program első részében lévő lépéseket, és jobbról balra mozgatják a labdát. A 90-es sor kicsit különbözik a 40-es sortól, mivel a labda ellentétes irányban mozog /jobbra kell letörölni a labdát és balra kell mozgatni/.

És mikor ez mind kész van, a program visszamegy a 20-as sorba, hogy ujrakezdje az egész eljárást. Elég tömör!

A program egy uj változata kedvéért, javitsa a 40-es sort a következőre:

```
40 PRINT "●";
```

a ● kiirásához tartsa lenyomva a SHIFT-et és nyomja meg a "Q" betüt.

Hajtsa végre a programot és nézze meg, most mi történik. Mivel kihagytuk a kurzor vezérlést, minden labada helyén marad a képernyőn, amig a program második részében, a jobbról balra mozgó labda le nem törli.

INPUT /bevitel/

Eddig minden beállitásra került a programon belül még mielőtt azt végrehajtottuk. Ha egyszer elinditottuk a programot, semmit nem lehetett változtatni. Az INPUT lehetővé teszi számunkra, hogy végrehajtás közben uj információt adjunk át a programnak és, hogy a program eszerint az uj információ szerint járjon el.

Ahhoz, hogy elképzelésünk legyen, hogy hogyan dolgozik az INPUT, gépelje be NEW *RETURN*, és irja be ezt a rövid programot:

```
10 INPUT A$
20 PRINT "AZT GEPELTE BE, HOGY: ";A$
30 PRINT
40 GOTO 10
RUN
? COMMODORE 64
AZT GEPELTE BE, HOGY: COMMODORE 64

? ■
```

ÖN IRTA

A GÉP VÁLASZOLT

E program végrehajtásakor egyszerü dolog történik. Egy kérdőjel fog megjelenni, jelezve, hogy a számitógép várja, hogy valamit begépeljen. Irjon be bármilyen karaktert, vagy karakter csoportot a billentyüzetről és nyomja meg a *RETURN*-*t*. A számitógép válasza "AZT GEPELTE BE, HOGY:" lesz, melyet az az információ fog kötni, amit beirt.

Ez nagyon kezdetlegesnek tünhet, de képzelje el mi min-

dent csinálhat bármely begépelt információval a számitógép.

Beolvashat akár numerikus akár füzér változókat, sőt egy üzenettel is tájékoztathatja az INPUT utasitás a felhasználót. Az INPUT formája ez:
INPUT " TAJEKOZTATO UZENET"; változó

az üzenetnek 40 karakternél rövidebbnek kell lennie

Vagy csak:
INPUT változó

MEGJEGYZÉS:

Ahhoz, hogy kikerüljön ebből a programból, nyomja meg a *RUN/STOP* és a *RESTORE* billentyüket.

A következő program nemcsak hasznos, hanem megmutat sok eddig ismertetett dolgot, beleértve az uj bevitели utasitást.

```
NEW
READY.
1 REM HOMERSEKLET KONVERTALO PROGRAM
5 PRINT"■" :REM "■ = CLR"
10 PRINT "FAHRENHEITBOL VAGY CELSIUSBOL"

20 PRINT "KONVERTALJON (F/C)";:INPUT A$
30 IF A$="F" THEN 100
40 IF A$<>"C" THEN 10
50 INPUT "ADJA MEG A CELSIUS FOKOT: ";C
60 F = (C*9)/5+32
70 PRINT C;"CELSIUS FOK = ";F;" FAHRENHE
IT FOK"
80 PRINT
90 GOTO 10
100 INPUT "ADJA MEG A FAHRENHEIT FOKOT:
";F
110 C = (F-32)*5/9
120 PRINT F;" FAHRENHEIT FOK = ";C;" CEL
SIUS FOK"
130 PRINT
140 GOTO 10
```

CSAK A SOR VÉGÉN NYOMJA LE A RETURN-T!

Ha beirja és végrehajtja ezt a programot, müködésben látja az INPUT-ot.

A 20-as sor nem csak információ összegyüjtésére használja a beolvasó utasitást, hanem a tájékoztatás kiirására is. Vegye észre azt is, hogy akár numerikus, akár füzér információt kérhet /azáltal, hogy numerikus vagy füzés változót ad meg/.

A 30 és 40-es sorok elemzik azt, amit begépelésre került. A 30-as sorban, ha F lett begépelve kiderül, hogy a felhasználó a Fahrenheit-fokban lévő hőmérsékletet akarja Celsius fokba átszámolni, igy a program arra a részre ugrik, ahol az ezirányu átszámitás van.

A 40-es sor még egy ellenőrzést végez. Tudjuk, hogy csak két érvényes választása van a felhasználónak. Ahhoz, hogy a 40-es sorba eljusson a program, a felhasználónak valami F-től különböző karaktert kellett beütnie. Itt egy ellenőrzés történik arra, hogy ez a karakter vajon C-e; ha nem, a program uj begépelést kér.

Ez mind részlet kérdésnek látszik, de jó programozási gyakorlat. Egy a programot nem ismerő felhasználót nagy kuuarc érhet, ha a program valami furcsát csinál, mert hibát követett el az információ beirásánál.

Ha egyszer meghatároztuk milyen átszámitást kell végrehajtani, a program elvégzi az átszámitást, és kiirja a beadott és az átszámolt hőmérsékletet.

Az átszámolás sima matematika, ami a hőmérséklet átszámolásra felállitott képletet használja. A számolás befejezése és a válasz kiirása után a program visszahurkol és ujra kezdődik.

A végrehajtás után a képernyő igy nézhet ki:

```
FAHRENHEITBOL VAGY CELSIUSBOL
KONVERTALJON (F/C)? F
ADJA MEG A FAHRENHEIT FOKOT: ? 32
 32  FAHRENHEIT FOK =  0  CELSIUS FOK

FAHRENHEITBOL VAGY CELSIUSBOL
KONVERTALJON (F/C)? ■
```

A program végrehajtása után győződjön meg róla, hogy kimentette lemezre vagy szalagra. Ez a program, ugyanugy mint a kézikönyvben bemutatott többi, programkönyvtára alapját képezheti.

GET /vesz/

A GET egyszerre egy karakter begépelését teszi lehetővé a billentyüzetről a *RETURN* megnyomása nélkül. Ez több felhasználásnál ténylegesen meggyorsitja az adatok begépelését. Bármikor megnyomunk egy billentyüt, az hozzárendelődik a GET utasitásban megadott változóhoz.

A következő rutin bemutatja, hogy hogyan müködik a GET utasitás:

```
NEW

READY.
1 PRINT "□" :REM "□ = CLR"
10 GET A$: IF A$ = "" THEN 10
20 PRINT A$
30 GOTO 10
```

Ha végrehajtja a programot, letörlődik a képernyő és valahányszor megnyom egy billentyüt, a 20-as sor ki fogja irni a képernyőre, és utána másik karaktert vesz. Nagyon fontos megjegyezni, hogy a beirt karakter nem jelenik meg a képernyőn, ha kifejezetten ki nem irja a PRINT utasitással a képernyőre,ahogy itt tettük.

A 10-es sorban lévő második utasitás szintén fontos. A GET folyamatosan dolgozik, akkor is ha nem nyomunk meg billentyüt /nem úgy mint az INPUT, amely egy válaszra vár/, igy e sor második része folyamatosan ellenőrzi a billentyüzetet, amig egy billentyüt meg nem nyomunk.

Nézze meg mi történik ha a 10-es sor második részét elhagyjuk.
A program leállitása végett a *RUN/STOP* és *RESTORE* billentyüket nyomja meg.

A hőmérséklet átszámitó program első részét könnyen át lehet irni, hogy GET-et használjon. Töltse be a hőmérséklet átszámitó programot LOAD-dal, és a következőképpen modositsa a 20-as és 40-es sorokat, valamint adja hozzá a 25-ös sort.

```
20 PRINT "KONVERTALJON (F/C)"
25 GET A$: IF A$ = "" THEN 25
30 IF A$="F" THEN 100
40 IF A$<>"C" THEN 25
```

Ez a modositás folyamatosabbá teszi a program müködését, mivel semmi sem fog történni addig, amig a felhasználó be nem gépeli az átszámolás tipusának kiválasztásához megkivánt válaszok egyikét.

Ha elvégezte a változtatást, győződjön meg róla, hogy elmentette a program uj változatát.

VÉLETLEN SZÁMOK ÉS MÁS FÜGGVÉNYEK

A Commodore 64 számos olyan függvénnyel rendelkezik, amelyeket különleges müveletek elvégzésére használunk. A függvényeket ugy tekinthetjük, mint a BASIC-be beletartozó, beépitett programokat. De ahelyett, hogy begépelne számos utasitást minden olyan esetben, amikor valamilyen különleges számolásra van szüksége, csak begépeli a kivánt tevékenységhez tartozó parancsot és a többit elvégzi a számitógép.

Valahányszor egy játék vagy oktató programot tervez, szüksége van véletlen számok előállitására, például ahhoz, hogy a kocka dobást szimulálja. Bizonyára tud irni egy olyan programot amely előállitja ezeket a számokat, de könnyebb módszer az RND /RaNDom/ véletlen szám függvény meghivása.

Hogy lássa, hogy ténylegesen mit csinál az RND, próbálja ki ezt a rövid kis programot:

```
NEW

READY.
10 FOR X= 1 TO 10
20 PRINT RND(1),
30 NEXT
```

A program végrehajtása után egy ilyen kijelzést fog látni:

```
.736410177          .573158057
4.69653672E-03      .330362232
.224685248          .951582529
.432380538          .971813238
.0683007652         .30226277
```

Hogy nem egyeznek a számok? Nos, ha ezt tennék, bajban lennénk, mivel teljesen véletleneknek kell lenniük! Próbálja meg még néhányszor ujra végrehajtani a programot, hogy meggyõzõdjön róla, hogy az eredmények mindig különbözõek. Ha a számok nem is követnek semmilyen sémát, elkezdheti megfigyelni, hogy néhány dolog ugyanolyan marad valahányszor végrehajtja a programot.

Elõször, az eredmény mindig 0 és 1 között van, de sosem egyenlõ 0-val vagy 1-gyel. Ez bizonyára sosem lesz megfelelõ, ha véletlen kocka dobásokat akarunk szimulálni, mivel akkor 1 és 6 közé eső számokat keresünk.

A másik lényeges jellegzetesség amit láthatunk, hogy valós számokkal (amelyek tizedeseket is tartalmaznak) foglalkozunk. Ez szintén proglémát jelenthet, mivel gyakran egész számokra van szükség.

Számos egyszerü módja van, hogy a kivánt határok közé eső számokat állitsunk elõ az RND függvénnyel.

Helyettesitse a 20-as sort a következõvel és ujra hajtsa végre a programot:

```
10 FOR X= 1 TO 10
20 PRINT 6*RND(1),
30 NEXT
RUN
 5.78635943          4.94914079
 1.71208391          2.07594889
 5.06752359          1.60849879
 3.35223833          2.92863922
 .624843367          4.21179364
```

Ez megoldotta azt a korábbi problémánkat, mely szerint nem kaptunk 1-nél nagyobb eredményt, de az értékek még mindig tartalmaznak tizedes részt. Most egy másik függvényt hivhatunk.
Az INTeger /egész rész/ függvény minden valós számot egész értékké alakit.

Mégegyszer helyettesitse a 20-as sort a következővel és hajtsa végre a programot, hogy lássa a változtatás hatását:

```
10 FOR X= 1 TO 10
20 PRINT INT(6*RND(1)),
30 NEXT
RUN
 3          2          5          3
 5          1          2          0
 5          4
```

Ez sok mindent megold, közelebb segitve minket ahhoz az eredeti célunkhoz, hogy 1 és 6 közé eső véletlen számokat hozzunk létre. Ha közelebbről megvizsgálja az utóljára létrehozott számokat, azt találja, hogy az eredmény csak 0 és 5 között van.

Utolsó lépésként adjon hozzá egyet az utasitáshoz, ahogy itt következik:

```
20  PRINT INT(6*RND(1))+1,
```

Nos, elértük a kivánt eredményt.
Általánosságban számot, változót vagy bármilyen BASIC

kifejezést elhelyezhet az INT függvény zárójelei között. A kivánt határoktól függően csak meg kell szoroznia az RND függvény felső határát. Például, 1 és 25 közötti véletlen számok létrehozásához, begépelheti:

```
20 PRINT INT (25*RND(1))+1
```

Az általános formula egy adott /nyilt,tehát a határokat is tartalmazó/ intervallumba eső véletlen számok előállitására:

szám=INT((felső határ-alsó határ+1)*RND(1))+alsó határ.

KITALÁLOS JÁTÉK

Mivel már elkövettünk néhány dolgot a véletlen számok megértése érdekében, miért ne használjuk fel ezt az információt. A következő játék nem csak a véletlen számok helyes használatát mutatja be, hanem néhány további programozási elvbe is bevezetést nyujt.

E program végrehajtásakor egy SZ véletlen szám jön létre.

```
 1 REM SZAM KITALALOS JATEK
 2 PRINT"J"
 5 INPUT"ADD MEG A FELSO HATART";HT
 10 SZ=INT(HT*RND(1))+1
 15 TG=0
 20 PRINT"MEGVAN A SZAMOM":PRINT
 30 INPUT"MI A TIPPED";TP
 35 TG=TG+1
 40 IF TP>SZ THEN PRINT"AZ EN SZAMOM KISE
BB":PRINT:GOTO30
 50 IF TP<SZ THEN PRINT"AZ EN SZAMOM NAGY
OBB":PRINT:GOTO30
 60 IF TP=SZ THEN PRINT"NAGYSZERU! ELTALA
LTAD"
 65 PRINT"OSSZESEN";TG;" PROBALKOZASSAL":
PRINT
 70 PRINT"JATSZOL MEG EGYET (I/N)?";
 80 GET VL$: IF VL$="" THEN 80
 90 IF VL$="I" THEN 2
 100 IF VL$<>"N"THEN 80
 110 END
```

A program kezdetén megadhatja, hogy a szám maximális értéke milyen nagy lehet. Ezután az Ön feladata, hogy kitalálja a számot.

```
ADD MEG A FELSO HATART? 25
MEGVAN A SZAMOM

MI A TIPPED? 15
AZ EN SZAMOM KISEBB

MI A TIPPED? 5
AZ EN SZAMOM NAGYOBB

MI A TIPPED? 8
NAGYSZERU! ELTALALTAD
OSSZESEN 3  PROBALKOZASSAL

JATSZOL MEG EGYET (I/N)?
```

Az IF...THEN utasitások összehasonlitják találgatását az előállitott számmal. Találgatásától függően, a program megmondja, hogy találgatása nagyobb vagy kisebb volt, mint az előállitott véletlen szám.

A véletlen számok terjedelmének meghatározására adott formulából láthatja, hogy hozzáadható néhány sor a programhoz, ami lehetővé teszi a felhasználó számára, hogy a véletlen számok terjedelmének alsó határát is megadja.
Valahányszor végez egy találgatást, TG 1-el növększik, nyomon követve a találgatások számát. A program használata során figyelje meg, hogy tud-e találni egy jó stratégiát a szám legkevesebb próbálgatással történő kitalálására.
Ha eltalálta a helyes választ, a program a "NAGYSZERU! ELTALALTAD" üzenetet irja ki, a próbálgatások számának kiséretében. Ezután ujra kezdheti az eljárást. Ne felejtse el, a program minden esetben uj véletlen számot állit elő.

PROGRAMOZÁSI TANÁCSOK:

A 40-es és 50-es sorban kettőspontot használtunk, az egyik sorba irt utasitások elválasztására. Ez nemcsak gépelést takarit meg, de hosszu programok esetében tár helyet is megtakarit.
Ugyanebben a két sorban figyelje meg, hogy az IF... THEN utasitásokban arra utasitottuk a számitógépet, hogy irjon ki valamit ahelyett, hogy közvetlenül a program valamilyen más pontjára ugrana.
Végül lássuk a sorok 10-esével történő számozása mögött rejlő okot: miután befejeztük a program irását, elhatároztuk, hogy hozzávesszük a találgatások számának meghatározásását. Mivel a sorszámok a helyesen meglévő sorszámok közé estek. Ezeknek az uj soroknak a program végére oly módon történő hozzáadásával, hogy a sorszámaik a már meglévőké közé megfelelően essen /15,35,65/, a program modositása könnyen elvégezhető volt.

Az ÖN DOBÁSA

A következő program két kocka dobását szimulálja. Használhatja önállóan, ahogy itt áll, vagy nagyobb játék részeként.

```
5 PRINT "SZERENCSET PROBALSZ?"
10 PRINT "PIROS KOCKA =";INT(6*RND(1))+1

20 PRINT "FEHER KOCKA =";INT(6*RND(1))+1

30 PRINT"NYOMD LE A SZOKOZ BILLENTYUT ES
Y UJABB  DOBASHOZ":PRINT
40 GET A$: IF A$ = "" THEN 40
50 IF A$ = CHR$(32) THEN 10
```

Szeretné kipróbálni szerencséjét?

A véletlen számokról, a BASIC-ról tanultak alapján, nézze meg végig tudja-e követni mi történik.

VÉLETLEN GRAFIKA

A véletlen számokhoz fűződő utolsó megjegyzésként, és bevezetésképpen a grafikai tervezésbe, szánjon rá egy kis időt, hogy gépelje be és hajtsa végre ezt az egyszerű kis programot.

```
10 PRINT ":)"
20 PRINT CHR$(205.5+RND(1));
40 GOTO20
```

Ahogy várhatta, a 20-as soré a kulcs szerep. Egy másik függvény, a CHR$ /CHaRacter string= karakter fűzér/, a 0-tól 255-ig terjedő szabványos kód számok alapján egy-egy karaktert állit elő. A Commodore 64 által kiirható minden karakter ilyen módon van kódolva /lásd az F függeléket/.

Egy karakter kódjának gyors megtalálása érdekében, csak gépelje be:

```
PRINT ASC("X")
```

ahol X az ellenőrizni kivánt karakter /ez bármilyen kiirható karakter lehet, beleértve a grafikus jeleket is/. A válasz a begépelt karakter kódja. Ahogy valószinüleg rájött, az "ASC" egy ujabb függvény, amely a begépelt karakter szabványos "ASCII" kódját adja vissza.

A következő begépelésével ki tudja irni a karaktert:

```
PRINT CHR$(X)
```

Ha megpróbálja begépelni:

```
PRINT CHR$(205);CHR$(206)
```

az M és N billentyük jobb oldali grafikus jeleit fogja látni. Ezt a két karaktert használja a program a labirintushoz.

A 205.5+RND(1) formulát használva a számitógép 205.5 és 206.5 közé eső véletlen számot állit elő. Ötven-

Ötven százalékos az esély, hogy a szám 206 alatt vagy felett lesz. A CHR$ figyelmen kivül hagyja a tört részeket, igy az esetek felénél a 205-ös kódu karakter és a maradék esetben a 206-os kódu karakter jelenik meg.

Ha kisérletezni szeretne ezzel a programmal, próbálja meg, hogy megváltoztatja a 205.5-t néhány tizedes hozzáadásával vagy kivonásával. Ez azt fogja eredményezni, hogy valamelyik karakter nagyobb eséllyel kerül kiválasztásra.

5. FEJEZET

FEJLETT SZÍN ÉS GRAFIKUS PARANCSOK

- Szín és grafika
- Színek kiírása
- Színek CHR$ kódjai
- PEEK és POKE utasítások
- Képernyő grafika
- További pattanó labdák

SZÍN ÉS GRAFIKA

Eddig a Commodore 64 okos számitási képességeinek egy részét vizsgáltuk meg. De az egyik legelbüvölőbb tulajdonsága a szint és grafikát elöállitó kiemelkedő képessége.
A "pattanó labda" és a "labirintus" programok kapcsán már láthatott gyors példát a grafikára. De ezek csak érintették mindazt a képességet, amire Ön utasithatja a gépet. Számos uj fogalmat vezetünk be ebben a fejezetben, hogy elmagyarázzuk a grafika és szín programozást, és hogy megmutassuk, hogyan lesz képes saját játékokat és fejlett animációt alkotni.

Mivel eddig a gép számolási képességeire összpontositottunk, minden eddig létrehozott kijelzés egyszinü volt /világoskék szöveg sötétkék háttéren és világos kék keret/.

Ebben a fejezetben meg fogjuk látni, hogy hogyan rendeljünk szint a programokhoz és hogyan vezéreljük azokat a furcsa grafikus jeleket a billentyüzeten.

SZÍNEK KIIRÁSA

Amint azt felfedezhettte, ha kipróbálta a szin beállitási tesztet az 1. fejezetben, a *CTRL* billentyü és valamelyik szin billentyü lenyomásával meg tudta változtatni a szöveg szineit. Ez kiválóan müködik közvetlen módban, de mi történik ha a szin változtatást a programjába szeretné beiktatni?

Mikor a "pattanó labda" programot megmutattuk, látta, hogy billentyü parancsokat, mint például a kurzor mozgatást, PRINT utasitásba iktathatunk. Hasonló módon, a szöveg szinének változtatását is beleveheti a programjába.
16 szöveg szint eljes skáláját használhatja. A *CTRL* és egy szám billentyü használatával a következő szinek férhetők hozzá:

1	2	3	4	5	6	7	8
fekete	fehér	piros	türkiz	lila	zöld	kék	sárga

Ha a megfelelő szám billentyükkel együtt lenyomva tartja a C= billentyüt, ugy további nyolc szin használható:

1	2	3	4	5	6	7	8
narancs	barna	vil. piros	sötét szürke	közép szürke	vil. zöld	vil. kék	világos szürke

Gépelje be, hogy NEW, és kisérletezzen a következővel. Tartsa lenyomva a *CTRL* billentyüt és egyidejüleg nyomja le az 1 billentyüt. Ezután, a *CTRL* billentyü nélkül nyomja meg az S billentyüt. Most mig ujra megnyomja a *CTRI* billentyüt egyidejüleg nyomja meg a 2 billentyüt. Engedje el a *CTRL* billentyüt és nyomja meg a 2 billentyüt. Menjen végig a számokon, a betükkel váltakozva és gépelje be a SZIVÁRVÁNY szót, ahogy itt áll:

```
10  PRINT "S Z I V A R V A N Y"
    CTRL  1 2 3 4 5 6 7 8 1 2

RUN

SZIVARVANY
```

Ahogyan a kiiró utasitásban a kurzor vezérlők az idézőjelek között, grafikus jelekként jelentek meg, a szin vezérlők is grafikus karakterekként jelentkeznek.

Az előző példában, amikor lenyomva tartotta a *CTRL*-t és begépelte a 3-t, egy £ jelent meg. A *CTRL* 7 egy ← kijelzést jelentett. Ily módon használva mindegyik szin vezérlő egyedi grafikus kódként kerül kijelzésre. A táblázat az összes kiirható szin vezérlő grafikus kódját mutatja.

BILLENTYU	SZIN	KIJELZES	BILLENTYU	SZIN	KIJELZES
CTRL 1	FEKETE	"■"	C= 1	NARANCS	"▒"
CTRL 2	FEHER	"▒"	C= 2	BARNA	"▒"
CTRL 3	PIROS	"▒"	C= 3	VIL.PIROS	"▒"
CTRL 4	TURKIZ	"▒"	C= 4	SOT.SZURKE	"▒"
CTRL 5	LILA	"▒"	C= 5	KOZ.SZURKE	"▒"
CTRL 6	ZOLD	"▒"	C= 6	VIL.ZOLD	"▒"
CTRL 7	KEK	"▒"	C= 7	VIL.KEK	"▒"
CTRL 8	SARGA	"▒"	C= 8	VIL.SZURKE	"▒"

Bár a PRINT utasitás kicsit furcsán néz ki a képernyõn, amikor végrehajtja a programot, csak a szöveg kerül megjelenitésre. És a kiiró utasitásban elhelyezett szin vezérlõk szerint automatikusan cserélni fogja a szineket.

Próbáljon ki néhány saját példát, bármilyen számu szin cserélésével egy PRINT utasitáson belül. Ne felejtse el azt sem, hogy egy másodlagos szöveg szin készletet is tud használni, ha megnyomja a Commodore billentyüt és valamelyik szám billentyüt.

Tanács:

Szin vagy mód /inverz/ változtató programok végrahajtása után észre fogja venni, hogy a "READY" kiirás és bármely további szöveg amit begépel, az utolsó szin vagy mód változással megegyező lesz. Ahhoz, hogy visszakerüljön rendes kijelzés módba, emlékezzen rá, hogy meg kell nyomnia a:
RUN/STOP és*RESTORE* billentyüket.

SZINEK CHR$ KÓDJAI

Vessen egy rövid pillantást az F függelékre, aztán térjen vissza ehhez a fejezethez.
Megfigyelhette az F függelékben a CHR$ kódokat végignézve, hogy minden szinnek /ugyanugy mint a legtöbb billentyüzet vezérlõknek, amilyen például a kurzor mozgatás/ egyedi kódja van. Ezeket a kódokat közvetlenül is kiirhatjuk, hogy ugyanazokat az eredményeket kapjuk, mint amikor a *CTRL*-t és a megfelelõ billentyüt gépeljük be egy PRINT utasitásba.

Például, próbálja ki ezt:

```
10 PRINT CHR$(147) : REM "CLR"
20 PRINT CHR$(30);"CHR$(30) MILYENRE VAL
TOZTAT?"
```

```
CHR$(30) MILYENRE VALTOZTAT?
```

A szövegnek most zöldnek kell lennie. Sok esetben a CHR$ függvény használata sokkal könnyebb, különösen ha kisérleteket akar folytatni a szinek változtatására. A következő programban egy másik megoldás található a szivárvány szineinek kiirására. Mivel számos hasonló sor található /40-110/, használja a szerkesztő billentyüket a sok gépelés megtakaritása érdekében. Nézze meg a lista után szereplő SZERKESZTÉSI MEGJEGYZÉSEK-et, hogy felfrissitse memoriájában a szerkesztési eljárásokat.

```
1 REM AUTOMATIKUS SZINES CSIKOK
5 PRINT CHR$(147) : REM CHR$(147)="CLR"
10 PRINT CHR$(18);"      ";:REM INVERZ SZOKOZ
20 SZIN = INT(8*RND(1))+1
30 ON SZIN GOTO 40,50,60,70,80,90,100,110
40 PRINT CHR$(5);: GOTO 10
50 PRINT CHR$(28);: GOTO 10
60 PRINT CHR$(30);: GOTO 10
70 PRINT CHR$(31);: GOTO 10
80 PRINT CHR$(144);: GOTO 10
90 PRINT CHR$(156);: GOTO 10
100 PRINT CHR$(158);: GOTO 10
110 PRINT CHR$(159);: GOTO 10
```

Gépelje be az 1-től 40-ig terjedő sorokat rendesen.

Képernyőjének igy kell kinéznie:

```
1 REM AUTOMATIKUS SZINES CSIKOK
5 PRINT CHR$(147) : REM CHR$(147)="CLR"
10 PRINT CHR$(18);"      ";:REM INVERZ SZ
OKOZ
20 SZIN = INT(8*RND(1))+1
30 ON SZIN GOTO 40,50,60,70,80,90,100,11
0
40 PRINT CHR$(5);: GOTO 10
```

A *CRSR-FEL* billentyüvel lehet a kurzort a 40-es sorba helyezni. Aztán javitsa át a 40-esből a 4-est 5-re. Következő müvelet, hogy a *CRSR-JOBBRA* billentyüvel mozgassa el a kurzort a CHR$ zárójelei között lévő 5-re. Nyomja meg a *SHIFT INST/DEL* billentyüket, hogy egy üres hely keletkezzen és gépeljen be '28'-at. Most nyomja meg a *RETURN*-t bárhol is legyen a kurzor a sorban..

A képernyőnek most igy kell kinéznie:

```
1 REM AUTOMATIKUS SZINES CSIKOK
5 PRINT CHR$(147) : REM CHR$(147)="CLR"
10 PRINT CHR$(18);"      ";:REM INVERZ SZ
OKOZ
20 SZIN = INT(8*RND(1))+1
30 ON SZIN GOTO 40,50,60,70,80,90,100,11
0
50 PRINT CHR$(28);: GOTO 10
```

Ne aggodjon. A 40-es sor még megvan. Listázza ki a programot a LIST paranccsal és meglátja. Ugyanilyen eljárás alkalmazásával, folytassa úgy, hogy mindig az utolsó sort módositsa egy uj sorszám és CHR$ kód átirásával egészen addig, amig az összes maradék sort be nem irta. Nézze, mi előre megmondtuk, a szerkesztő billentyük még jól fognak jönni. Végső ellenőrzésképpen, irassa ki az egész programot, hogy mielőtt elinditaná, meggyőződjön róla, hogy minden sort helyesen gépelt be.

Egy rövid magyarázatot adunk, hogy mi fog történni. Valószinüleg elképzelte már a szín csík program legnagyobb részét, kivéve talán azt a furcsa uj utasitást a 30-as sorban. De nézzük gyorsan tulajdonképpen mit csinál az egész program. Az 5-ös sor kiirja a törlés /CLR/ CHR$ kódját.

A 10-es sor átvált inverz módba és kiir 5 szóközt, melyek, mint kiderült, csikot alkotnak, mivel inverz módban vagyunk. Először végrehajtva, a csik világoskék lesz, rendes szöveg szinü.

A 20-as sor használja a fő munkaerőnket, a véletlen

szám függvényt, hogy kiválasszon egy véletlen színt 1 és 8 között.

A 30-as sor az IF...THEN utasitások változatait tartalmazza, melyet ON...GOTO[1]-nak hivunk. Az ON...GOTO lehetővé teszi, hogy a program a sorszámok listájából válassza ki, hogy hova ugorjon. Ha a változó /ebben az esetben a SZIN/ értéke 1, az első sorszámot /itt a 40-est/ választja ki. Ha az értéke 2, a második sorszámot választja a felsorolásból és igy tovább.

A 40-110 között lévő utasitások épp a véletlen szín számainkat alakitják át a színnek megfelelő CHR$ kódra és visszaküldik a programot a 10-es sorba, hogy PRINT-el kiirjon egy csik részletet ezzel a szinnel. Aztán az egész eljárás elölről kezdődik.

Nézze meg, ki tudja-e találni, hogy hogyan lehet előállitani 16 véletlen számot, terjessze ki az ON...GOTO utasitást, hogy kezelje ezeket is, és tegye hozzá a többi CHR$ kódot, hogy a fennmaradt 8 színt is megjelenitse.

PEEK /les/ ÉS POKE /bök/ UTASITÁSOK

Nem, nincs szó a számitógép döfködéséről, de lehetőségünk lesz "benézni" a számitógép belsejébe és arra, hogy "beledugjunk" dolgokat.

Épp ugy, mint ahogy a változókat úgy tekintheti,mint képviselőit a gépben lévő olyan"dobozok"-nak, ahova az információt helyezte, ugyanigy elképzelhet néhány különlegesen értelmezett "dobozt" a számitógépben, amelyek adott célu tár helyeket képviselnek.

A Commodore 64 megnézi ezeket a tár helyeket, hogy lássa, milyen legyen a képernyő hátterének és keretének a szine, milyen karaktereket - és hol - kell megjeleníteni a képernyőn, és egy sereg más feladatnál.

A megfelelő tár helyre különböző értéket helyezve a POKE utasitással meg tudjuk változtatni a szineket, kialakitani és mozgatni tudunk tárgyakat, és még zenét is tudunk szerezni.

Ezeket a tár helyeket igy tudjuk ábrázolni:

1/ ON...GOTO itt: ha...menj

53280	53281	53282	53283
X	Y		
KERET SZIN	HÁTTÉR SZIN		

Az imént csak négy helyet mutattunk meg, melyek közül kettő a képernyő és a háttér szinét vezérli. Próbálja begépelni ezt:

```
POKE 53281,7  RETURN
```

A képernyő háttér szine sárgára változik, mert 7-es - sárga - értéket tettünk arra a helyre, ahol a képernyő háttér színének a vezérlése történik.

Próbáljon meg POKE-al különböző értékeket elhelyezni a háttér színt vezérlő helyre, és figyelje meg a kapott eredményeket. Bármilyen O és 255 közé eső értéket elhelyezhet,de csak a O és 15 közöttiek lesznek hatásosak.

Az egyes szinek valóságos értékei a POKE-al történő beíráshoz:

O	fekete	8	narancs
1	fehér	9	barna
2	piros	10	világos piros
3	türkiz	11	sötétszürke
4	lila	12	közép szürke
5	zöld	13	világos zöld
6	kék	14	világos kék
7	sárga	15	világos szürke

Ki tud találni egy módszert, hogy megjelenitse a különböző háttér és keret kombinációkat? A következő program nyujthat némi segitséget:

```
10 FOR HT = 0 TO 15
20 FOR KE = 0 TO 15
30 POKE 53280, HT
40 POKE 53281, KE
50 FOR X = 1 TO 200: NEXT X
60 NEXT KE: NEXT HT
```

Két egyszerü ciklust hoztunk létre, hogy POKE-al beirjuk a háttér és keret szinek változtatásához szükséges különböző értékeket. A KÉSLELTETŐ ciklus az 50-es sorban egy kicsit lelassitja a dolgokat.

A kiváncsiság kedvéért, próbálja ki ezt:

```
? PEEK (53280) AND 15
```

15-ös értéket kell kapnia. Ez a KERET-nek adott utolsó érték és érthető, mivel a program végrehajtása után mind a háttér mind a keret szürke szinü /15-ös érték/.

Az AND 15 beirásával figyelmen kivül hagyott 0-15 kivételével minden más értéket, ami a szín kódok számitógépbeli tárolási módja következtében elöállhatott. Rendesen ugyanazt az értéket várhatja, amit utoljára POKE-al beirt a helyre. Általában, a PEEK lehetővé teszi, hogy megvizsgáljunk egy előirt helyet és megnézzük milyen érték van jelenleg ott. El tudja képzelni egy olyan sor hozzáadását a programhoz, ami megjeleniti a program végrehajtása során a HÁTTÉR és KERET értékét?

Mit szól ehhez:

```
25 PRINTCHR$(147);"KERET = ";PEEK(53280) AND 15,
   "HATTER = ";PEEK(53281) AND 15
```

KÉPERNYŐ GRAFIKA

Az eddig végrehajtott információ kiirások mindegyikében a számitógép rendesen, sorban kezelte az információt:

egyik karaktert a másik után írta ki, a kurzor aktuális helyétől kezdve /kivéve, ha azt kérte, hogy új sort kezdjen, vagy a ';'-t használta a PRINT-ben formázásra/

Ahhoz, hogy egy adott helyre írjon ki PRINT-el először, azt tudja csinálni, hogy elindul a képernyő egy ismert helyéről és a képernyő formázásának megfelelő számú kurzor vezérlőt ír ki PRINT-el. De ez program lépéseket igényel és fogyasztja az időt.

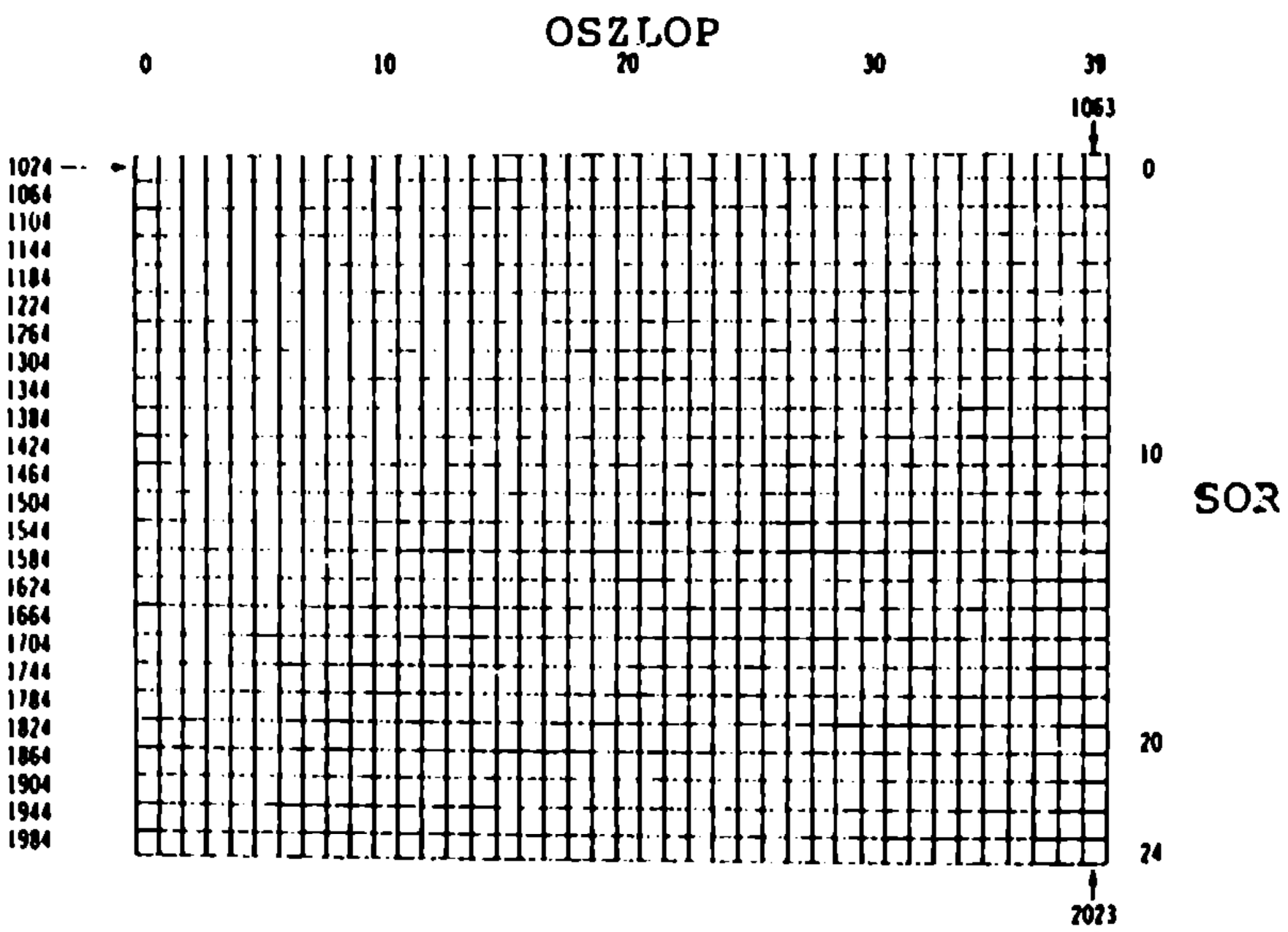
Azonban épp úgy, mint ahogy a szín vezérléséhez vannak bizonyos helyek a Commodore 64 tárjában, ugyanúgy vannak helyek, melyeket a képernyő egyes helyeinek közvetlen vezérléséhez tud használni.

KÉPERNYŐ TÁR TÉRKÉP

Mivel a számítógép képernyője 1000 karakter /40 oszlop, 25 sor/ ábrázolására képes, 1000 memória hely van fenntartva a képernyőn megjelenő információk kezelésére. A képernyő elrendezése rácsként képzelhető el, és mindegyik négyzet egy tár helyet képvisel.

És mivel a tárban minden egyes hely egy 0 és 255 közé eső számot tartalmazhat, minden egyes tár helynek 256 különböző értéke lehet. Ezek az értékek képviselik a Commodore 64 által megjeleníthető különböző karaktereket /lásd az E függeléket/. Egy karakternek tartozó értéknek a megfelelő képernyő tár helyre POKE-al történő beírásával a karakter a megfelelő helyen kerül kijelzésre.

A Commodore 64-nél a képernyő tár általában az 1024-es tár helyen kezdődik és a 2023-on végződik. Az 1024-es hely a képernyőnek a bal felső sarka. Az 1025-ös az ettől jobbra eső következő hely, és így tovább végig a az sorban. Az 1063-as hely a jobb szélső hely az első sorban. Egy sor utolsó karakterét követő következő hely az alatta lévő sor első karaktere.



Mondjuk, egy a képernyőn pattanó labdát szeretne vezérelni. A labada a képernyő közepén van, a 20. oszlopban és a 12. sorban. A képernyőn lévő hely memoria helyének kiszámitásához tartozó formula:

PONT HELYE = 1024 +X +40*Y

X — oszlop
Y — sor

ahol X az oszlop és Y a sor.

Igy a labda memoria helye: oszlop

1024 + 20 + 480 vagyis 1524

480 — sor (40*12)

Törölje le a képernyőt a ***SHIFT*** és ***CLR/HOME*** billentyükkel, és gépelje be:

```
POKE 1524,81
```

81 — karakterkód
1524 — hely

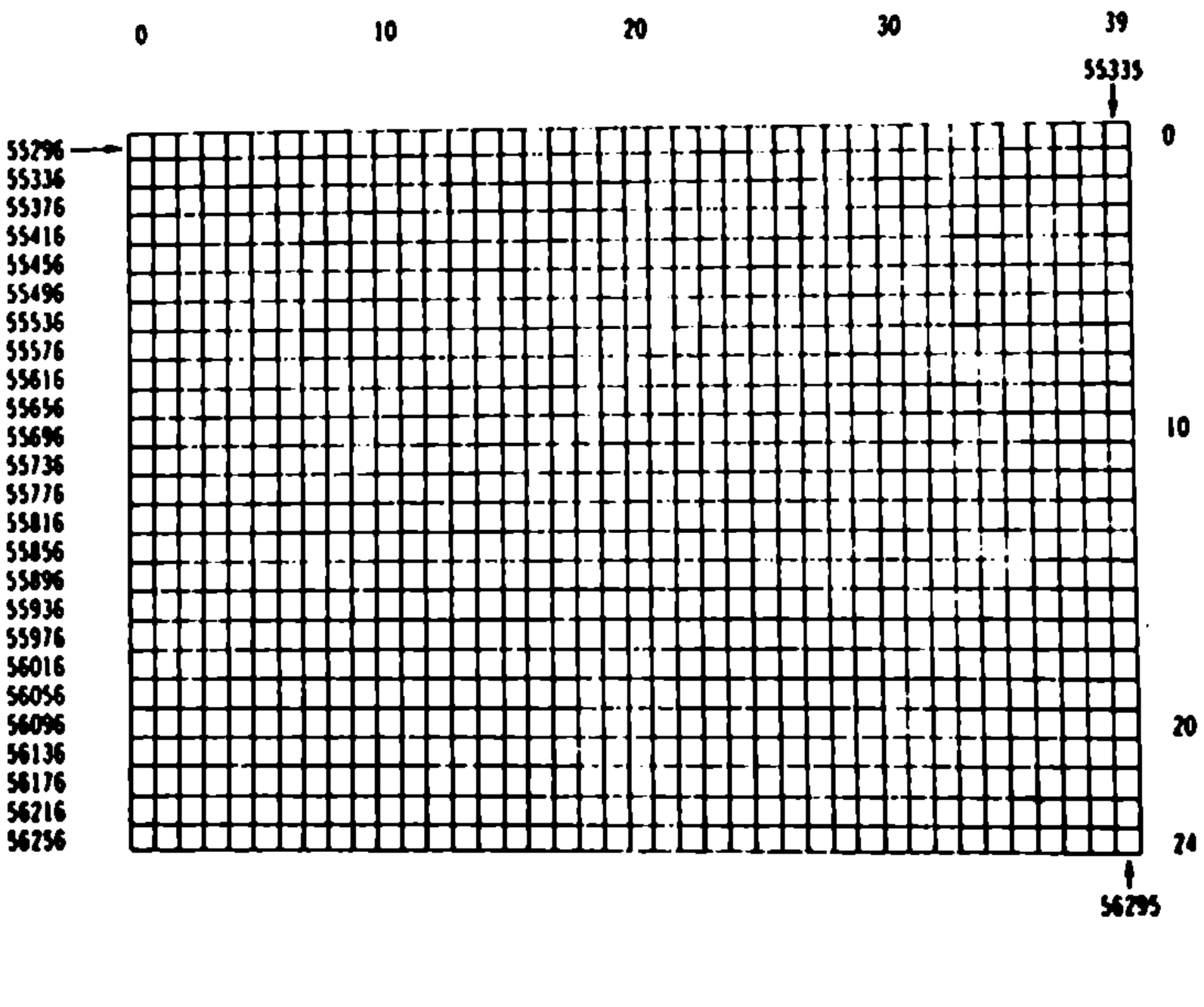
KÓSZÓ ZSOLT

SZÍN TÁR TÉRKÉP

Egy labda jelenik meg a képernyő közepén! PRINT utasitás használata nélkül közvetlenül elhelyezett egy karaktert a képernyő memoriába. A megjelenő labda fehér volt. Azonban egy másik tár terület megváltoztatásával arra is mód nyilik, hogy a képernyőn megjelenő tárgy színét megváltoztassuk. Gépelje be:

```
POKE 55796,2
     hely
          szin
```

A labda színe pirosra változik. A Commodore 64 képernyőjének minden helyéhez két tár hely tartozik, az egyik a karakter kód részére, a másik a szín kód részére. A szín tár térkép az 55296-os cimen kezdődik /ez a bal felső sarok/, és 1000 helyen keresztül folytatódik. Ugyanazokat a szín kódokat



használhatjuk 0 és 15 között itt a karakter színének változtatásához, mint amiket a keret és háttér színének kicseréléséhez használtunk.

A képernyõ tár helyek kiszámitásához használt formulát módosithatjuk úgy, hogy megadja a szín kódok POKE-al történõ beirásához szükséges helyeket.

Az uj formula a következõ:

PONT SZINE = 55296 + X + 40*Y

vagy másképp 54272-t kell a megfelelõ képernyõ helyhez hozzáadni.

TOVÁBBI PATTANÓ LABDÁK

Itt egy bõvitett pattanó labda program, amely közvetlenül POKE utasitásokkal ir a képernyõre ahelyett, hogy kurzor vezérlõket használna a PRINT utasitásokban. Amint a program végrehajtása után látni fogja, ez sokkal rugalmasabb mint a korábbi program, és elvezet minket sokkal kifinomultabb animációk programozásához.

```
10 PRINT"J"
20 POKE 53280,7 : POKE 53281,13
30 X = 1 : Y = 1
40 DX = 1 : DY = 1
50 POKE 1024 + X + 40*Y,81
60 FOR T = 1 TO 10 : NEXT
70 POKE 1024 + X + 40*Y,32
80 X = X + DX
90 IF X = 0 OR X = 39 THEN DX = -DX
100 Y = Y + DY
110 IF Y = 0 OR Y = 24 THEN DY = -DY
120 GOTO 50
```

A 10-es sor letörli a képernyõt, és a 20-as sor beállitja a hátteret világos zöldre, a keretet sárgára. A 30-as sorban lévõ X és Y változók a labda aktuális sor és oszlop helyét követik nyomon. A 40-es sorban, a DX és DY változók a labda mozgásának vizszintes és függõleges irányát adják. Amikor +1-t adunk az X-hez, a labda jobbra mozog; amikor -1 -t adunk hozzá, a labda balra mozog. +1 hozzáadása az Y-hez egy sorral lejjebb mozgatja a labdát; -1 hozzáadása az Y-hoz egy sorral feljebb mozgatja a labdát.

Az 50-es sor az aktuális kurzor helyre teszi a labdát a képernyőn. A 60-as sor a már ismerős késleltető ciklus, mely éppen annyi ideig hagyja a labdát a képernyőn, hogy látható legyen.

A 70-es sor letörli a labdát azáltal, hogy egy helyközt /32-es kódot/ ir arra a helyre, ahol a képernyőn a labda volt.

A 80-as sor hozzáadja az irány tényezőt az X-hez. A 90-es sor vizsgálja, hogy a labda eléri-e valamelyik oldal falat, és megforditja az irányt, ha visszapattanás következik be. A 100-as és 110-es sorok ugyanezt végzik el a felső és alsó falakra.

A 120-as sor visszaküldi a programot, hogy ujra megjelenitse és mozgassa a labdát.
Az 50-es sorban a 81-es kód valamilyen más kódra történő lecserélésével akármilyen más karakterre le tudja cserélni a labdát. Ha a DX-t vagy DY-t 0-ra változtatja, a labda egyenesen fog mozogni az átlós helyett.

Adhatunk hozzá még egy kicsit több intelligenciát. Eddig X és Y értékét csak azért ellenőriztük, hogy a képernyő határokon belül maradjunk. Vegye hozzá a következő sorokat a programhoz:

```
21 FOR L = 1 TO 10
25 POKE 1024 + INT(RND(1)*1000), 102
27 NEXT L
115 IF PEEK(1024+X+40*Y)=102 THEN DX=-DX:GOTO 80
```

POKE-KAL BEÍRANDÓ KÓD

A 21-től 27-ig terjedő sorok 10 blokkot helyeznek el a képernyőn véletlen helyekre. A 115-ös sor ellenőrzi /PEEK-el/, hogy a labda nekiütközik-e egy blokknak, és ha igen megváltoztatja az irányát.

6. FEJEZET

SZELLEM GRAFIKA

- Bevezetés a szellem grafikába
- Szellemek alkotása
- További megjegyzések a szellemekhez
- Bináris aritmetika

BEVEZETÉS A SZELLEM /SPRITE/

GRAFIKÁBA

Az elõzõ, grafikával foglalkozó fejezetekben láttuk, hogy animációk megalkotásához és grafikonszerü képek megjelenitéséhez a PRINT utasitásokban használhatunk grafikus jeleket.

Arra is mutattunk módszert, hogy POKE utasitással karakter kódokat helyezzünk el egy adott képernyõtár helyen. Ez aztán közvetlenül elhelyezi a megfelelõ karaktereket a képernyõn a helyes helyre.

Az animáció megalkotása mindkét esetben sok munkát igényel, mivel a tárgyakat meglévõ grafikus jelekbõl kell megalkotni. A tárgyak mozgatása számos program utasitást igényel, hogy nyomon követhessük, és uj helyre helyezhessük a tárgyat. Továbbá a korlátozott számu grafikus jel használata miatt a tárgy alakja és felbontása nem lehet olyan jó, mint ahogy azt szeretnénk.

A szellemek használata az animációkban kiküszöböli a problémák többségét. A szellem egy nagy felbontásu tárgy, ami majdnem bármilyen alakura formálható — BASIC parancsok segitségével. A tárgy könnyen mozgatható a képernyõn csupán azáltal, hogy közöljük a számitógéppel a helyet, ahová mozgatni akarjuk. A számitógép elvégzi a többit.

A szellemeknek még ennél is több képességük van. Változtatni lehet a szinüket; és meg tudja mondani, ha az egyik tárgy összeütközik a másikkal; az egyik elhaladhat a másik elõtt vagy mögött; és könnyen meg lehet növelni a méretüket, a kezdõk kedvéért.

Mind ennek az ára csekély. Azonban a szellemek használata még néhány további ismeretet igényel a Commodore 64 müködéséről, és arról, hogy a gépen belül, hogyan kezeli a számokat. Ez nem olyan nehéz, mint ahogyan hangzik. Csak kövesse a példákat, és meg fogja alkotni saját szellemjeit, melyek igen rövid idõ alatt bámulatos dolgokat fognak csinálni.

SZELLEMEK ALKOTÁSA

A Commodore 64 a szellemeket különálló képalkotóval vezérli. Ez a képalkotó kezeli a kép kijelzõt. Ez végzi el a karakterek és grafikus jelek nyomon követésébõl és megalkotásából, szinek létrehozásából, és a tárgyak mozgatásából eredõ nehéz munka egészét.

Ennek a kijelzõ áramkörnek 46 különböző "BE/KI" helye van, melyek mint belsõ tár helyek müködnek. Ezeknek a helyeknek mindegyike 8 blokkból álló sorozatra oszlik. És minden egyes blokk akár "be" akár "ki" állapotu lehet. Erről késõbb adunk részletesebb információt. A megfelelõ decimális értéknek a POKE utasitással a helyes tár helyre történő irásával vezérelni tudja szellem alkotásainak elhelyezését és mozgását.

A kép megalkotásához szükséges tár helyekhez történő hozzáférésen felül, néhány fõtárbeli helyet is használni fogunk a szellemeket meghatározó információk /adatok/ tárolásához. Végül, közvetlenül a képernyõ tárrész után lévõ nyolc tár helyet fogjuk arra használni, hogy megmondjuk a számitógépnek, hogy az egyes szellemekhez pontosan melyik tár területrõl vegye az adatokat.

Ahogy végignézünk néhány példát, az eljárás nagyon egyszerü lesz, és bele fog tanulni a kezelésbe.

Nos, haladjunk elõre néhány szellem grafika megalkotásával. Egy szellem objektum 24 pont széles és 21 pont magas. Egyidejüleg legfeljebb nyolc szellem vezérelhetõ. A szellemek egy különleges nagy felbontásu módban kerülnek kijelzésre, ami a képernyõt 320 pont széles és 200 pont magas területre osztja.

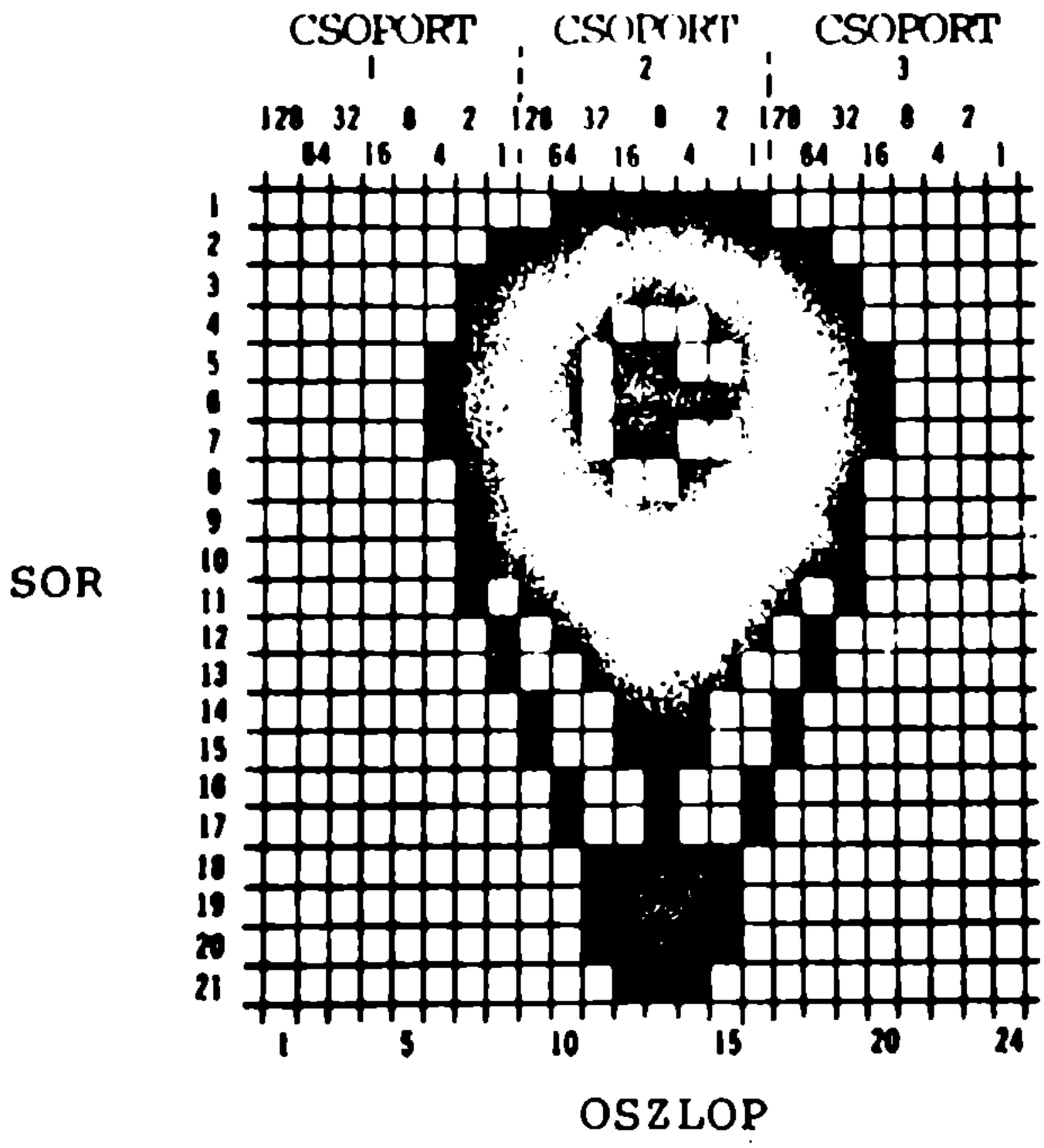
Mondjuk, egy léggömböt akar létrehozni és lebegtetni akarja az égbolton. A léggömböt egy 24-szer 21-es rácson tudja megtervezni /lásd az ábrán/.

A következõ lépés, hogy a megtervezett rajzot a számitógép számára használható adatokká alakitsa át. Vegyen egy darab papirt és vegyen fel egy lefelé 21, keresztben 24 egység hosszu minta rácsot. A tetejére keresztben irja fel a következõt, 128,64,32,16,8,4,2,1, háromszor egymás után /lásd az ábrát/, hogy mind a 24

négyzet fölé kerüljön szám. Számozza meg a rács baloldalán a sorokat 1-től 21-ig. Most töltse ki a rácsot bármilyen elgondolás szerint,vagy használja a léggömbünket. A legegyszerübb elõször körvonalazni az alakot és utána visszatérni a rács kitöltésére.

Gondolja azt a besötétitett négyzetekről, hogy azok "be" állapotuak és minden besötétitett négyzetet helyettesitsen 1-el. Azok, melyek nincsenek kitöltve, a "ki" állapotuak, ezek helyére nullát tegyen.

Az elsõ sorral kezdve, a pontokat át kell számolnia három különálló adat részbe, amit a számitógép el tud olvasni. Mindegyik, 8 négyzetből álló csoport egy adat részt nek felel meg a léggömbünkben, melyet byte-nak [1] hivunk. Balról kezdve, az elsõ 8 négyzet üres, vagyis 0, igy e számsorozat értéke 0.



1/ Ejtsd: bájt. Helyesebb elnevezése az oktet

A középső sorozat igy néz ki /ujra: az 1 egy pontot jelöl, a 0 egy üres helyet/:

128	64	32	16	8	4	2	1
0	1	1	1	1	1	1	1

0 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127

Az első sor harmadik sorozata ugyancsak üres, igy ez is nullával egyenlő. Igy az első sorhoz tartozó adat:1/

```
DATA  0,127,0
```

A második sorhoz tartozó sorozatokat igy számolhatjuk:

1 sorozat:	0	0	0	0	0	0	0	1

1 = 1

2 sorozat:	1	1	1	1	1	1	1	1

128+64 +32 +16 + 8 +4 + 2 + 1 = 255

3 sorozat:	1	1	0	0	0	0	0	0

128+64 = 192

A 2-es sorhoz tartozó adat:

```
DATA 1,255,192
```

Ugyanilyen módon lehet a fennmaradó sorokat alkotó hármas sorozatokat decimális értékre átszámolni. Forditson rá időt és végezze el a maradék átszámitásokat ebben a példában.

Most, hogy az objektumhoz szükséges minden adat rendelkezésre áll, hogyan lehet ezt felhasználni? Gépelje be a következő programot és nézze meg mi történik.

Ha mindent helyesen gépelt be, léggömbje simán száll keresztben az égen.

1/ Adat angolul 'data'

```
1 REM FEL, FEL, ES ELORE
5 PRINT "J"
10 V=53248: REM VIDEO IC KEZDOCIME
11 POKE V+21,4: REM SZELLEM 2 ENGEDELYEZESE
12 POKE 2042,13: REM SZELLEM 2 ADATAI A 13.BLOKKBOL
20 FOR N=0 TO 62: READ Q: POKE 832+N,Q: NEXT
30 FOR X=0 TO 200
40 POKE V+4,X: REM X KOORDINATAK ATIRASA
50 POKE V+5,X: REM Y KOORDINATAK ATIRASA
60 NEXT X
70 GOTO 30
200 DATA 0,127,0,1,255,192,3,255,224,3,227,224
210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
230 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

Q-nak értéket ad DATA adatokból*

ADATOK, melyeket READ ad át Q-nak*

* Részletesebben lásd READ és DATA a 8. fejezetben

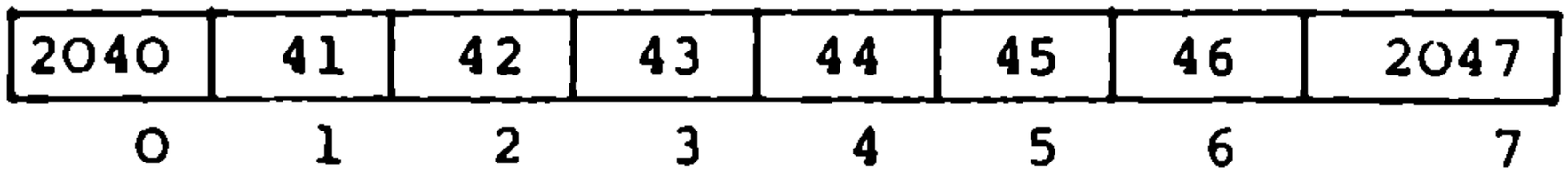
Hogy megértse mi történt, elõször tudnia kell, hogy mely kép létrehozó helyek vezérlik az Önnek szükséges tevékenységeket. Ezeket a helyeket regiszteknek hivjuk,1/ és igy lehet õket ismertetni:

Regiszter	Leirás
0	0-ás szellem X koordinátája
1	0-ás szellem Y koordinátája
2-15	0-áshoz hasonló párok az 1-7 szellemekhez
16	legfelsõ bit az X koordinátához
21	szellem engedélyezése:1= megjelenik 0= eltünik
29	a szellem megnövelése "X" irányában
23	a szellem megnövelése "Y" irányban
39-46	0-7 szellemek szinei

Ezen információn tulmenõen azt is meg kell mondania minden szellemhez, hogy hol vannak a hozzátartozó adatok.

Ezeket az adatokat közvetlenül a képernyõtárat követõ 8 helyen kezeljük:

1/ A regisztereket ugyanugy kezeljük, mint eddig a tár helyeket



Vázoljuk a dolgok mozgatásához szükséges pontos eljárást és végül irjunk egy programot.

Csak néhány dolog szükséges ahhoz, hogy tényleg megalkossunk és mozgassunk egy tárgyat.

1. Tegye a megfelelő szelleme/ke/t láthatóvá a képernyőn, azáltal, hogy a 21-es helyre POKE utasitással megfelelő számot tesz, ami bekapcsolja a szelleme-/ke/t.

2. Állitsa be a szellem mutatókat /2040-7 tárhelyek/ arra a cimre, ahonnét a megfelelő szellemhez tartozó adatot ki kell olvasni.

3. POKE utasitással helyezze el a tényleges adatot a tárba.

4. Egy ciklus segitségével, változtassa az X és Y koordinátákat, a szellem mozgatásához.

5. Szabadon választhat még, hogy kivánja-e kiterjeszteni a tárgy méretét, változtatni a színét, vagy végre kiván hajtani különféle különleges müveletet.

A 29-es hely használatával megnövelheti a szellemét "X" irányban és a 23-as hellyel "Y" irányban.
Az eddig elmondottak után már csak néhány olyan elem lehet a programban, ami nem ismerős.

A 10-es sorban:

V = 53248

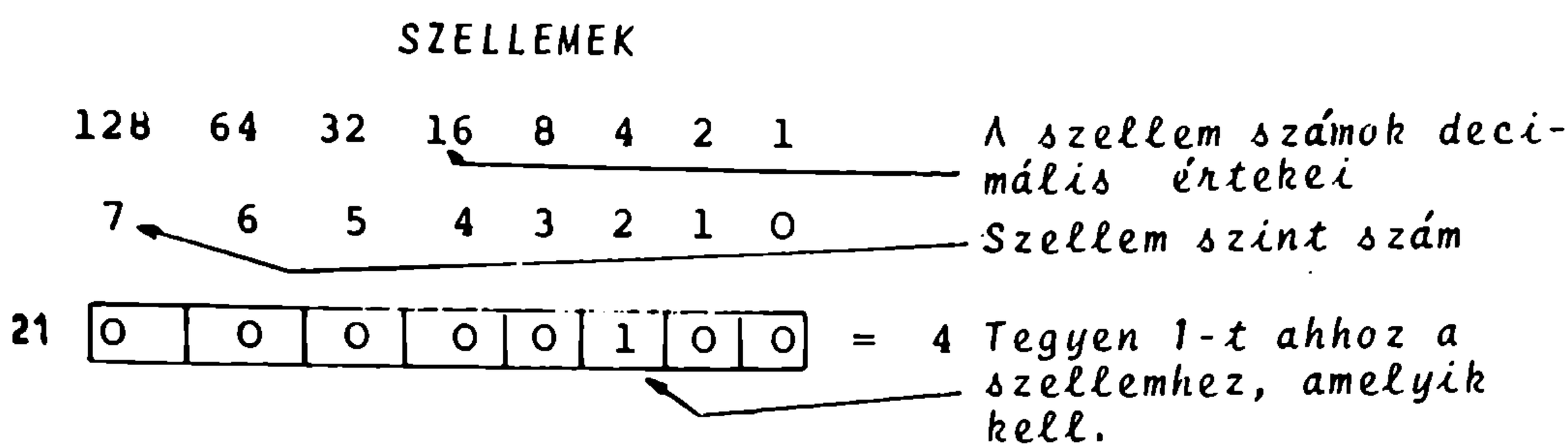
beállitja V-be a képalkotó áramkör kezdő tár helyét.[1/]
Ezzel a módszerrel csak a V-t kell növelni a tár számmal ahhoz, hogy megkapjuk az aktuális tár helyet. A regiszter számok azok, melyeket a regiszter térképen megadtunk.

A 11-es sorban:

1/ Ez éppen a 0-s regiszternek felel meg

```
POKE V + 21,4
```

láthatóvá teszi a 2-es szellemet azáltal, hogy 4-t tesz a 21-es engedélyező regiszterbe, ami a 2-es szellem bekapcsolására alkalmas. Ilyesmire gondoljon:



Mindegyik szellem szint ábrázolásra kerül a szellem tár 21-es helyén és a 4-es érték felel meg a 2-es szintü szellemnek. Ha a 3-as szintü szellemet szeretné használni, a 3-as szellemhez kellene 1-t tenni, melynek 8 az értéke. Sõt, ha a 2-es és a 3-as szellemet is használná, mind a 4-es mind a 8-as helyre 1-t kellene tennie. Aztán össze kellene adni a számokat ugy mint a rajzpapirján az adatokkal tette. Igy a 2-es és 3-as sprite bekapcsolását a V+21,12 végezné el.

A 12-es sorban:

```
POKE 2042,13
```

arra utasitja a számitógépet, hogy a 2-es szellemhez /2042-es hely/ tartozó adatot a tár 13. területérõl vegye. A létrehozott szellemjéből tudja, hogy az 63 tár helyet foglal el. Lehet, hogy nem vette észre, de azok a számok amiket a rács tetejére irt, a számitógép 3 bytejának felelnek meg. Más szóval, a következõ számok minden egyes csoportja, 128,64,32,16,8,4,2,1, a számitógép tárában egy-egy byte-nyi helyet foglal el. Igy rácsának 21 sora szorozva minden sorhoz tartozó 3 byte-al, minden szellem esetében 63 byte-nyi tárat foglal el.

```
20  FOR N = 0 to 62: READ Q: POKE 832-N,Q: NEXT
```

1 egész szellem

Ez a sor végzi az aktuális szellem megalkotást. A szellemet képviselő 63 byte adatot egy ciklusban POKE utasitással betesszük a tár 13. blokkjába. Ez a 832-es helyen /13*64/ kezdődik.1/

```
30 FOR X = 0  TO 200
40 POKE V +4, X
50 POKE V +5, X
```

2-es sprite X koordinataja

2-es sprite Y koordinataja

Ha emlékszik rá az iskolából, egy tárgy X koordinátája vizszintes irányu helyzetét, és az Y koordinátája a szellem függőleges helyét mutatja. Igy mivel X értékei 0-tól 200-ig változnak a 30-as sorban /egyesével/ a szellem egy hellyel LE és JOBBRA mozog a képernyőn minden következő számnál. A számokat a számitógép elég gyorsan olvassa ahhoz, hogy a mozgás folyamatosnak tünjön, ahelyett, hogy lépésenként látszana. Ha több részletre van szüksége vessen egy pillantást az O függelékben lévő regiszter térképre.

Ha egyszerre több tárgyat kezd el mozgatni, egyetlen regiszter pár esetén ez nem lenne lehetséges, mivel mind a nyolc tárgy helyét egyszerre módositaná. Ezért minden egyes szellem 2 saját tár hellyel rendelkezik, hogy mozgatni lehessen a képernyőn.

A 70-es sor ujrakezdi a ciklust, miután a léggömb egyszer elhaladt a képernyőn. A program maradék része a léggömbhöz tartozó adat. Bizonyára másképp mutat a képernyőn, igaz?

Próbálja meg hozzátenni a következő sort:

```
25 POKE V + 23,4 : POKE V + 29,4: REM KITERJESZTES
```

és hajtsa végre ujra a programot. A léggömb az eredeti

1/ Mélyebb ismeretek nélkül a 13., 14. és 15. blokk használható.

mérete kétszeresére nőtt! Amit tettünk, egyszerü volt. POKE utasitással 4-t /ujra a 2-es szellemi jelzésképpen/ tettünk a 23 és 29-es tár helyekre, és a 2-es szellemet kiterjesztettük X és Y irányban.

Fontos megjegyezni, hogy a szellem a szellem blokk bal felsö sarkában kezdödik. Ha bármelyik irányban kiterjesztünk egy tárgyat, a kezdö pont ugyanaz marad.

További izgalom kedvéért végezze el a következö változtatásokat:

```
11 POKE V+21,12

12 POKE 2042,13; POKE 2043,13

30 FOR X = 1 TO 190

45 POKE V +6,X

55 POKE V +7,190-X
```

Egy második /3-as számu/ szellemet kapcsoltunk be azáltal, hogy POKE utasitással 12-t tettünk a sprite megjelenését engedélyezö tár helyére /V+21/. A 12-es érték a 2-es és 3-as sprite-okat kapcsolja be /00001100=12/ [1]

A hozzáadott 45-ös és 55-ös sorok mozgatják a 3-as szellemet azáltal, hogy POKE utasitással értékeket adnak a 3-as szellem X és Y koordináta helyeire /V+6 és V+7/.

Még több cselekményt szeretne a képernyöre? Próbálja meg ezeket a kiegészitéseket:

28 = 4 +8 +16 → A 2:,3., és 4. SZELLEM

```
11 POKE V+21,28

12 POKE 2042,13: POKE 2043,13: POKE 2044,13

13 REM MEGADJA MINDEN SZELLEMHEZ HONNET VEGYE AZ
   ADATOT

25 POKE V+23,12: POKE V+29,12: REM MEGNOVELI A 2-ES
   ES 3-AS SZELLEMET
```

1/ Lásd a BINÁRIS ARITMETIKA fejezetet

```
48 POKE V+ 8, X
58 POKE V+ 9,100
```

Ezúttal a 11-es sorban egy ujabb szellemet jelenitettünk meg azáltal, hogy 28-at tettünk a szellem tár szakasz megfelelő helyére. Most már a 2-4 szellemnek vannak bekapcsolva /00011100= 28/.

A 12-es sor azt jelzi, hogy a 4-es szellem ugyanarról a tár helyről fogja venni az adatokat /a 13. 63 byte-os területről/, mint a többi szellem, ez a POKE 2044,13 eredménye.

A 25-ös sorban 12-t elhelyezve /2 és 3-as szellem beállitása/ az X és Y irányu kiterjesztési tár helyekre /V+23 és V+29/, a 2-es és 3-as szellemet megnöveltük.

A 48-as sor az X tengely mentén mozgatja a 3-as szellemet. Az 58-as sor a képernyő közepére, a 100-as sorba helyezi a 3-as szellemet. Mivel ez az érték nem változik, ellentétben azzal ahogy eddig változott X = 0-tól 200-ig, a 3-as szellem csak vizszintesen mozog.

TOVÁBBI MEGJEGYZÉSEK A SZELLEMEKHEZ

Most, hogy már kisérletezett a szellemekkel, ejtsünk róluk néhány további szót. Először az eddig a karakterek szinének meghatározásához használt 16 szabványos szin kód bármelyikére meg tudja változtatni egy szellem szinét. Ezeket megtalálhatja az 5. fejezetben vagy a G függelékben.
Például ahhoz, hogy az 1-es szellem szinét világos zöldre változtassa, gépelje be: POKE V+40,13 /győződjön meg róla, hogy V = 53248-t beállitotta/.

Észrevehette a szellem mintaprogramok használatánál,hogy a tárgy sosem ment el a képernyő jobb széléig. Ez azért volt, mert a képernyő 320 pont széles és az X irányu regiszter csak legfeljebb 255 nagyságu számot tud tárolni. Akkor hogyan tud egy tárgyat a képernyő teljes szélességében mozgatni?

Van a tár térképen egy hely amit még nem emlitettünk. A 16-os hely vezérel valamit, amit a szellem X irányu helyéhez tartozó legfelső /legmagasabb helyiértékü/ bitnek /MSB/ neveztünk. Valójában, ez teszi lehetővé, hogy a

szellem a vizszintes 256 és 320-as helyek között mozogjon.

Az X regiszter MBS-je igy müködik: miután a szellemet elmozgatta az X irányu 255-ös helyig, helyezzen el egy értéket a mozgatni kivánt szellemnek megfelelően a 16-os tár helyre. Például, ahhoz a 2-es szellemet vizsintesen mozgatni tudja a 256-320 helyeken, POKE utasitással helyezze el a 2-es szellemhez tartozó értéket, amely 4, a 16-os tár helyre:

```
POKE V+16,4
```

Most kezdjen el ujra 1-től elindulva értékeket elhelyezni a szokásos módon a 2-es szellemekhez tartozó X irányu regiszterbe /amely a tár térkép 4-es helye/. Mivel csak további 64 hellyel mozgatja odább, az X helyek csak 0 és 63 közötti tartományban mozognak.

Ezt az elvet legjobban az eredeti, első szellem program egy változatával lehet bemutatni:

```
5 PRINT"J"
10 V=53248 :POKE V+21,4 :POKE 2042,13
20 FOR N=0 TO 62:READ Q:POKE 832+N,Q:NEXT
25 POKE V+5,100
30 FORX=1TO255
40 POKEV+4,X
50 NEXT
60 POKE V+16,4
70 FOR X=0 TO 63
80 POKE V+4,X
90 NEXT
100 POKE V+16,0
110 GOTO30
200 DATA0,127,0,1,255,192,3,255,224,3,227,224
210 DATA7,217,240,7,223,240,7,217,240,3,231,224
220 DATA3,255,224,3,255,224,2,255,160,1,127,64
230 DATA1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA0,62,0,0,62,0,0,62,0,0,28,0
```

A 60-as beállitja a legfelső bitet a 2-es szellemhez. A 70-es sorban kezdődik az X irányu helyekhez tartozó értékek betöltése, mely az ut hátralévő részén mozgatja a képernyőn a 2-es szellemet.

A 100-as sor azért lényeges, mert lekapcsolja az MBS-t, igy a sprite ujra a képernyő bal széléről kezdheti a mozgását.

BINÁRIS ARITMETIKA

E bevezető kézikönyvnek nem feladata, hogy részletes információt adjon arról, hogy hogyan kezeli a számitógép a számokat. Azonban jó alapokat adunk az eljárás megértéséhez, a kifinomult animációba való bevezetése céljából.

De mielőtt belebonyolódnánk, definiálnunk kell néhány fogalmat:

BIT -- Ez a legkisebb terjedelmü információ, amit a számitógép tárolni tud. Ugy gondoljon a bitre, mint egy kapcsolóra, amely vagy "be" vagy "ki" állapotu lehet. Amikor egy BIT "be" állapotu, értéke 1; amikor "ki", értéke 0.

A BIT után következő szint a BYTE.

BYTE -- Ezt ugy definiáljuk, mint BIT-ek sorozatát. Mivel egy BYTE 8 BIT-ből áll, valójában a BIT-ek 255 különféle kombinációját jelenti. Más szóval, lehet az összes BIT-je "ki" állapotu, ekkor a BYTE-ja igy néz ki:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	0

és az értéke 0 lesz. Minden BIT "be" állapotu lehet:

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

ami 128+64+32+16+8+4+2+1=255

A következőt REGISZTER-nek hivjuk.

REGISZTER -- Ugy definiáljuk, mint blokkot alkotó BYTE-ok sorát. De ebben az esetben mindenegyes REGISZTER valójában csak 1 BYTE hosszu. A REGISZTER-ek sorozata egy REGISZTER TÉRKÉPET alkot. A REGISZTER TÉRKÉP-ek olyan elrendezések, mint amilyeneket a LÉGGÖMB SZELLEM-nél látott. Mindegyik REGISZTER különböző müveletet vezérel, például a SZELLEM-et bekapcsolót valójában ENGEDÉLYEZŐ REGISZTER-nek hivtuk. A SZELLEM-et hosszában megnövelőt X KITERJESZTÉS REGISZTER-nek, a SZELLEM-et széltében megnövelőt Y KITERJESZTÉS REGISZTER-nek. Jegyezze meg, hogy a REGISZTER egy olyan BYTE, amely különleges feladatot lát el.

Nézzük a BINÁRIS ARITMETIKA többi részét.

ÁTSZÁMOLÁS BINÁRISROL DECIMÁLISRA

DECIMÁLIS ÉRTÉK								
128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	1	2↑0
0	0	0	0	0	0	1	0	2↑1
0	0	0	0	0	1	0	0	2↑2
0	0	0	0	1	0	0	0	2↑3
0	0	0	1	0	0	0	0	2↑4
0	0	1	0	0	0	0	0	2↑5
0	1	0	0	0	0	0	0	2↑6
1	0	0	0	0	0	0	0	2↑7

Mind a nyolc bit kombinációját felhasználva, bármilyen értéket előállithat O-tól 255-ig. Kezdi már érteni, hogy miért kellett az értékeknek O és 255 közé esniök, amikor karakter vagy szin értékeket helyeztünk a tár helyekre POKE utasitással? Minden tár hely egy byte információt tud tárolni.

Nyolc O-nak és 1-nek bármilyen kombinációját át lehet számolni egy egyedi, O-255 közé esö decimális értékké. Ha minden helyen 1 van, akkor a byte értéke 255-el

egyenlő. Ha mind nulla, a byte értéke nullával egyenlő; " OOOOOOll" 3-al egyenlő, és igy tovább. Ez lesz az alapja olyan adatok létrehozásának, melyek szellemeket ábrázolnak és kezdelnek. Épp úgy, mint az egyik példában, ahol ez a byte csoportositás egy szellem egy részét ábrázolta /O üres hely, l kitöltött terület/:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	1	1	1	1	1

128 +64 + 32 + 16 + 8 + 4 + 2 + 1 = 255

Aztán POKE utasitással 255-t kell a megfelelő tár helyre tenni, hogy a tárgynak ezt a részét ábrázoljuk.

Tanács:

Hogy megtakaritsa a bináris számok decimálisra történő átszámolásánál fellépő gondokat -- ezt sokszor kell végrehajtanunk -- a következő program elvégzi ezt a munkát Ön helyett. Hasznos ötlet a program begépelése és elmentése későbbi felhasználás céljából.

```
5 REM BINARISROL DECIMALISRA KONVERTALO
10 INPUT"A 8 BITES BINARIS SZAM: ";A$
12 IF LEN(A$)<>8 THEN PRINT"8 BITET, LEGYSZIVES...":GOTO10
15 DE = 0 : H = 0
20 FOR X=8 TO 1 STEP -1:H=H+1
30 DE = DE + VAL(MID$(A$,H,1))*2↑(X-1)
40 NEXT X
50 PRINT "BINARIS ";A$;" = ";"DECIMALIS ";DE
60 GOTO 10
```

A program beveszi a bináris számot, melyet füzérként adunk meg, és végignézi a füzér minden karakterét, balról jobbra /a MID$ függvénnyel/. A H változó jelzi, hogy éppen melyik karakterrel dolgozik a program a ciklusban.

A 30-as sorban lévő VAL függvény a karakter tényleges

értékét adja vissza. Amig szám karakterekkel dolgozunk, az érték ugyanaz, mint a karakter. Például, ha az A$ első karaktere 1, az érték szintén 1 lesz.

A 30-as sor befejező része a szóbanforgó karakter értékét megszorozza 2 megfelelő hatványával. Mivel a példában az első érték, a 2 ↑ 7-en helyen, 1-es, a DE először 1-szer 128-al, azaz 128-cal egyenlő. Ha a bit 0, akkor az erre a helyre kerülő érték is 0.

Ezt az eljárást ismételjük mind a nyolc karakterre, és a DE nyomon követi a bináris szám egymást követő decimális értékeit.

7. FEJEZET

HANG LÉTREHOZÁSA

- Hanghasználat, ha ön nem számitógép programozó
- A hangkészitő program szerkezete
- Minta hang program
- Zenélés a COMMODORE 64-n
- Fontos hang beállitások
- Egy dallam lejátszása a COMMODORE 64-n
- Hanghatások létrehozása
- Hanghatás minták kipróbáláshoz

HANGHASZNÁLAT HA ÖN NEM SZÁMITÓGÉP "PROGRAMOZÓ"

A legtöbb programozó kétféle céllal használja a számitógép hangját: zenél vagy hanghatásokat hoz létre. Mielõtt belemennénk a hang programozásának bonyodalmaiba, nézzük meg gyorsan egy jellemzõ hangkészitõ program szerkezetét... és adjunk egy rövid hangprogramot, amelylyel Ön kisérletezhet.

A HANGKÉSZITÕ PROGRAM SZERKEZETE

Kezdjük azzal, hogy 5 féle beállitandó érték van amelyet Önnek ismernie kell ahhoz, hogy hangot hozzon létre a COMMODORE 64-en: HANGERÕ, HULLÁMALAK, FELFUTÁS/LECSENGÉS, TARTÁS/ELENGEDÉS/utóbbi négy jellemzõt FLTEnek röviditjük/ és MAGAS FREKVENCIA /ALACSONY FREKVENCIA. Az elsõ négy értéket rendszerint EGYETLENEGYSZER, a program elején állitjuk be. A magas: és alacsony frekvencia beállitást MINDENEGYES HANGJEGYRE el kell végezni. Ime ilyen egy tipikus hang/zene program szerkezete.

MINTA HANG PROGRAM

Indulás elõtt válasszon ki egy SZÓLAMOT. 3 szólam lehet. Mindenegyes szólam különbözõ hangbeállitási értékeket kiván a hullámalakhoz, stb. Ön 1, 2 vagy 3 hangot is megszólaltathat egyidejüleg, de a mintánk csak az 1.SZÁMU SZÓLAMOT használja. Gépelje be a következõ programot sorról sorra... ne felejtse el minden sor után leütni a *RETURN*-t:

1. a legnagyobb HANGERÕ beállitása:

```
10 POKE 54296,15
```

2. a FELFUTÁS/LECSENGÉS értékek beállitása annak meghatározására, hogy a hang milyen gyorsan emelkedjen

a maximális hangerõig ill. onnan milyen gyorsan essen vissza a TARTÁS szintjéig /0-tól 255-ig/:

```
20 POKE 54277,190
```

3. a TARTÁS/ELENGEDÉS értékeinek beállitása a hangnak egy adott hangerõn való meghosszabbitása, majd elengedése céljából:

```
30 POKE 54278,298
```

4. az M függelékben levõ zenei hang táblázatból kikeresett, játszani kivánt hanghoz tartozó MAGAS FREKVENCIA ÉS ALACSONY-FREKVENCIA értékek beállitása /minden egyes hanghoz 2 POKE szükséges/:

```
40 POKE 24273,17: POKE 54272,37
```

5. a HULLÁMALAK beállitása a 4 szabványos lehetõség közül az egyikre /17,33,65 vagy 129/:

```
50 POKE 54276,17
```

6. egy idõzitõ ciklussal beállitjuk a hang TARTAMÁNAK értékét /egy negyedhang körülbelül "250", de ez változhat mivel egy hosszabb program befolyásolja az idõzitést/:

```
60 FOR T = 1 TO 250: NEXT
```

7. végül a HULLÁMALAK és az FLTE beállitások kikapcsolása következik:

```
70 POKE 54276,0 :POKE 54277,0:POKE 54278,0
```

A létrehozott hang meghallgatásához gépelje be a RUN-t és üsse le a *RETURN*-t. Ha látni akarja, vagy változtatni akar a programon, akkor irja be a LIST szót és üsse le a *RETURN* billentyüt.

ZENÉLÉS A COMMODORE 64-EN

Nem kell, hogy zenész legyen ahhoz, hogy zenéljen a COMMODORE 64-en. Csak azt a néhány egyszerü számot kell ismernie, amelyik megmondja a számitógépnek, hogy

milyen hangosra állitsa a hangerõt, hogy melyik hangot játssza és milyen hosszan, stb. De elõször lássuk az alábbi programot, amely egy gyors bemutatást ad a COMMODORE 64 hihetetlen zenei lehetõségeirõl, EGYETLEN szólamot használva a lehetséges 3 független szólamból.

Gépelje be a NEW-t és üsse be a *RETURN*-t az elõzõ program törlése céljából, ezután vigye be az alábbi programot, majd RUN és*RETURN* beütésével futassa le /a programlista után minden egyes sorhoz magyarázatot füzünk/:

```
5 REM ZENEI SKALA
10 POKE 54296,15
20 POKE 54277,9
30 POKE 54276,17
40 FOR T=1TO300:NEXT
50 READ A
60 READ B
70 IF B=-1 THEN POKE 54273,0:POKE 54272,0:END
80 POKE 54273,A:POKE 54272,B
90 POKE 54276,0
100 GOTO20
110 DATA 17,37,19,63,21,154,22,227
120 DATA 25,177,28,214,32,94,34,175
900 DATA -1,-1
```

5 : program neve

10 : a hangerõ beállitása a legnagyobbra /15/

20 : felfutás/lecsengés beállitása /minden egyes hangra vonatkozik/

30 : hullámalak meghatározás /hangfajta/

40 : minden egyes hang tartama /hangzásának hossza/

50 : a 110-s DATA sorból beolvassa az elsõ értéket

60 : a 110-s DATA sorból beolvassa a második értéket

70 : a program "kikapcsolja" a hangot a két frekvencia beállitás nullázásával és leáll /END/ ha a READ a 900-s sorból /-1/-t olvas be

80 : a 110-s sor adataiból az első értéket /A=17/ beirja POKE-kal a MAGASABB FREKVENCIAKÉNT, mig a második számot /B=37/ ALACSONY FREKVENCIAKÉNT. A következő ciklusmenetben A = 19 és B = 63, stb. és ezeket irja be POKE-kal a MAGAS ÉS ALACSONY FREKVENCIA tárhelyekre. Az 54273 tárhely az 1. szólam MAGAS FREKVENCIÁJA van és az 54272 az 1. szólam ALACSONY FREKVENCIÁJA.

90 : a hullámforma beállitás kikapcsolása

100 : uj hangjegy beállitásra és lejátszásra megy vissza

110 : az M függelékben levő táblázatból kiválasztott zenei hang értékekek. Minden egyes számpár egy-egy hangjegyet reprezentál. Pl. 17 és 37 adja a 4. oktávbeli "C" hangot; 19 és 63 a "D"-t stb.

900 : amikor a program eléri a /-1/-t, akkor a MAGAS/ALACSONY FREKVENCIA beállitást kikapcsolja és END-del véget ér , ahogyan a 70. sor erre utasitja

Ahhoz, hogy "csemballó"-ra cserélje a hangot, a 30-as sort változtassa a következőre:

```
30 POKE 54276,33
```

és igy futtassuk le ujra a programot. /A sorcserét ugy hajtjuk végre, hogy a program leállása után - amit a READY kiirás jelet -, ujrairjuk a programnak azt a sorát, amelyet ki akarunk cserélni; az uj sor automatikusan a régi helyére kerül/. Amit most csináltunk, az a "háromszög" hullámalaknak a cseréje "fürészfog" formájura. A HULLÁMALAK csere jelentősen megváltoztatja a COMMODORE 64 által előállitott hangot, de a hullámalak csak az egyike a különböző lehetséges beállitásoknak, amelyeknek változtatásával különböző hangzásu zenei hangokat és hanghatásokat állithatunk elő. Ugyancsak megváltoztathatja minden egyes hang FELFUTÁS/LECSENGÉS gyorsaságát, pl. ahhoz, hogy a "csemballó" hang inkább "bendzsó" jellegü legyen, próbáljuk a 20-as sort kicserélni az alábbira:

```
20 POKE 54277,3
```

Mint láthatta, a COMMODORE 64 különböző hangszerek hangjait képes előállitani. Végezetül nézzük meg közelebbről, hogy az egyes hang beállitások mit eredményeznek.

KÓS20' 25OLT

FONTOS HANG BEÁLLITÁSOK

1. Hangerõ

- a hangerõ bekapcsolásához és legnagyobb hangerõre való beállitásához gépeljük be: POKE 54296,15.
A beállitható érték O-tól 15-ig terjed, de a 15-t fogja leginkább használni. A hangerõ "kikapcsolható" a következõ módon:

```
POKE 54296,0.
```

A hangerõ értéket csak EGYSZER kell a program elején beállitani, mivel ugyanaz a beállitás érvényes a Commodore 64 mindhárom szólamára./Zenei hang vagy hanghatás hangzása alatti hangerõ változtatás érdekes eredményt adhat, de ez már meghaladja a bevezetés hatókörét/.

2. Hullámalak beállitás

Látta már, hogy a hullámalak megváltoztatása hogyan változtatja meg a hang szinezetét "xilofon"-ról "csembaló"-vá. Mindegyik SZÓLAM rendelkezik saját HULLÁMALAK BEÁLLITÁSSAL, amely négy különböző tipusú hullámalakot határozhat meg: háromszög, fürészfog, impulzus /négyszög/ és zaj.
Egy minta hullámalak beállitás a következõképpen néz ki:

```
POKE 54276,17
```

ahol az elsõ szám /54276/ megadja az ELSÕ SZÓLAMRA a hullámalak beállitás tárhelyét és a második szám /17/ adja meg a háromszög hullámalakot. Az egyes SZÓLAMOK és HULLÁMALAKOK kombinációit láthatjuk a következõ táblázatban.

HULLÁMALAK BEÁLLITÁSOK

	tárhely	háromszög	fürészfog	négyszög	zaj
1. Szólam	54276	17	33	65	129
2. Szólam	54283	17	33	65	129
3. Szólam	54290	17	33	65	129

Noha a tárhelyek különböznek a szólamok szerint, az egyes hullámforma beállitási értékek azonosak ugyanazon hullámalakra. Hogy megértsük ennek müködését, tekintsük a hangskála program 20. sorát. E programban közvetlenül a HANGERÕNEK a 10. sorban való beállitása után az 1. SZÓLAM HULLÁMALAK BEÁLLITÁSÁT adtuk meg a 20. sorban, beirván POKE-kal az 54276. tárhelyre 17-et. Ez bekapcsolta az 1. SZÓLAMOT és beállitott egy HÁROMSZÖG hullámalakot /17/. Késõbb a hullámalakot 17 helyett 33-ra állitottuk fürész fog hullámalak elõállitására és igy csembalószerü hangskálát kaptunk. Figyeljük meg a hangerõ beállitás és a hullámalak beállitás különbségét. A hullámalak beállitása a hangerõ beállitáshoz hasonló, de minden szólamhoz külön beállitás tartozik és hangerõ szintek bevitele helyett hullámformákat adunk meg. Most lássuk a hang egy másik aspektusát, az FLTE lehetõséget.

3. Felfutás/lecsengés beállitás

A HULLÁMALAK BEÁLLITÁS mellett minden SZÓLAM-ra külön-külön meghatározható az FLTE, vagyis a COMMODORE 64 FELFUTÁS/LECSENGÉS/TARTÁS/ELENGEDÉS lehetõsége.

Kezdetnek nézzük meg a FELFUTÁS/LECSENGÉS beállitását. A következõ táblázat megadja a különbözõ FELFUTÁS és LECSENGÉS értékeket az egyes szólamokra. Ha a hang felfutás és lecsengés fogalmakkal nincs tisztában, akkor a "felfutást" ugy képzelje el, mint azt az idõt, amely alatt a hang a MAXIMÁLIS HANGERÕRE emelkedik. A LECSENGÉS pedig az az idõ, amely alatt a hang a maximális hangerõrõl a TARTÁS szintjére megy vissza.

A köetkezõ táblázat a FELFUTÁS/LECSENGÉS beállitást adja meg minden szólamra. A FELFUTÁS és LECSENGÉS beállitás értékeit ÖSSZE LEHET KOMBINÁLNI úgy, hogy összeadjuk õket és az összeget irjuk be. Pl. Ön beállithat egy NAGY FELFUTÁS és egy ALACSONY LECSENGÉS értéket, ugy hogy nagy felfutást /64/ és a kis lecsengést /1/ összeadjuk.
A/65/ érték mondja a számitógépnek, hogy nagy felfutást és kis lecsengést állitson be.
A felfutás idejét meg lehet növelni az egyes felfutási értékek összeadásával.

/128+64+32+16 = max. 240/.

FELFUTÁS/LECSENGÉS BEÁLLITÁSOK

	tár-hely	nagy fel-fu-tás	köze-pes fel-futás	kis fel-fu-tás	leq-ki-sebb fel-fu-tás	nagy le-csen-gés	köze-pes-le-csen-gés	kis-le-csen-gés	leg-kisebb lecsen-gés
1 szólam	54277	128	64	32	16	8	4	2	1
2 szólam	54284	128	64	32	16	8	4	2	1
3 szólam	54291	128	64	32	16	8	4	2	1

Ha felfutást lecsengés nélkül ad meg, akkor a lecsengés automatikusan nulla, és forditva. Pl.

POKE 54277,64

a közepes felfutás értéket nulla értékü lecsengéssel állit be az 1. SZÓLAMNÁL. Ha

POKE 54277,66-t

ir, akkor ez közepes felfutást és kis lecsengést eredményez /mert 66= 64 + 2/.
Összeadhat különböző felfutási értékeket vagy különböző lecsengési értékeket is. Pl. összeadhat egy kis felfutást /32/ és egy közepes felfutást /64/ egy 96 értékü felfutás érdekében, ezután hozzáadhat egy közepes lecsengési értéket /4/ és ... gyorsan irja be... POKE 54277,100.

Ennél a pontnál célszerü lesz egy programmal bemutatni a hatást. Gépelje be a NEW-t és üsse be a *RETURN*-t, RUN-nal futassa le a programot /a program listája után megtalálja az egyes sorok magyarázatát/:

```
10 PRINT"USSON LE EGY BILLENTYUT"
20 GETB$:IFB$=""THEN20
25 POKE54276,0:POKE54277,0:POKE54272,0
30 POKE54296,15
40 POKE54277,64
50 POKE54276,17
60 POKE54273,17:POKE54272,37
90 GOTO20
```

10: képernyõ üzenet

20: billentyüzet ellenõrzése

25: beállitások kikapcsolása

30: hangerõ beállitása a legnagyobbra

40: felfutás/lecsengés beállitása

50: hullám forma beállitás /háromszög/

60: egy hangot ir be az 1. szólamba

90: vissza a ciklus elejére, ujra csinálja.

Itt az 1. SZÓLAMOT használtuk egyidejüleg egy hang megszólaltatására, KÖZEPES FELFUTÁSSAL és NULLA LECSENGÉSSEL. A 40-s sor itt a kulcs. 64-t beirva POKE-al a FELFUTÁS/LECSENGÉS tárhelyére közepes felfutást hozott létre.
Az eredmény ugy hangzik, mintha egy olajos hordóban labda pattogna. Most egy érdekes részhez értünk. Üsse be a *RUN/STOP*-*t* a program leállitásához, azután gépelje be azt, hogy LIST és a *RETURN*-t nyomja le. Most gépelje be a következõ sort és utána a *RETURN*-t nyomja le /az uj 40-s sor automatikusan felülirja a régi 40-s sort/:

```
40 POKE 54277,190
```

Gépelje be azt, hogy RUN és üsse be a *RETURN*-*t*, hogy meghallgathassa, hogyan szól. Amit tettünk, az a különbözõ felfutás és lecsengés beállitások kombinációja.
A beállitás a következõ: NAGY FELFUTÁS /128/+ALACSONY

FELFUTÁS /32/ + KÖZEPES LECSENGÉS /4/ + KIS LECSENGÉS /2/ = 190. Ez a hatás ugy hangzik, mintha egy oboa vagy más "fuvola" jellegü hangszer szólalna meg. Ha kisérletezni szeretne, próbálja ugy változtatni a hullámforma és a felfutás/lecsengés számokat a hangskála példában, hogy oboa hangot kapjon. Igy láthatja, hogy a felfutás/lecsengés értékek megváltoztatásával hogyan lehet különböző tipusu hanghatásokat kapni.

4. A tartás/elengedés beállitása

Ugyanugy mint a felfutás/lecsengés beállitása, a TARTÁS/ELENGEDÉS beállitása is része a FLTE/HULLÁMFORMA vezérlésnek.

A TARTÁS/ELENGEDÉS lehetővé teszi egy adott hang egy részletének "megnyujtását" /TARTÁS-át/ mintha a zongora pedálját vagy az orgona billentyüit használná egy hangjegy meghosszabbitására . Bármely hang vagy hangzás a TARTÁS értékével meghatározott hangerőn tartható - sőt ha a maximális /240/ értékkel adta meg a tartás szintjét elengedés nélkül, a hang a "végtelenségig" fog szólni a maximális hangerővel. A TARTÁS/ELENGEDÉS beállitása egy FOR... NEXT ciklussal együtt használható annak meghatározására, hogy milyen hosszan legyen a tartási szinten tartva a hang. A következő táblázat megmutatja azokat a számokat, amelyeket beirva POKE-al a tárba különböző TARTÓS/ELENGEDÉS értékekhez jutunk.

TARTÁS/ELENGEDÉS BEÁLLITÁSOK

	tár-hely	hagy-tar-tás	közé-pes tar-tás	ki-csi tar-tás	leg-ki-sebb tar-tás	nagy elen-gedés	kö-ze-pes el-eng.	ki-csi el-eng.	leg-kisebb elen-gedés
1.szólam	54278	128	64	32	16	8	4	2	1
2.szólam	54285	128	64	32	16	8	4	2	1
3.szólam	54292	128	64	32	16	8	4	2	1

Például, ha az 1. szólamot használjuk, akkor egy magas tartási szintü beállitást kaphatunk a POKE 54278,128-al, vagy összekombinálhatjuk a magas tartást a kis elengedéssel a 128+2 összeadásával, ekkor POKE 54278, 130-t kell megadni.

Itt ugyanazt az egyszerü programot adtuk meg mint a FELFUTÁS/LECSENGÉS résznél,de hozzátettük a TARTÁS/ ELENGEDÉS lehetõséget is.

Figyeljük meg a hangzásbeli különbséget.

A 45. sorban megmondjuk a számitógépnek, hogy nagy szinten tartsa a hangot /128-t veszünk a fenti táblázatból/. A tartás szintjét megváltoztathatja a 40. sorban megadott "szám" változtatásával, mint ahogy a 40. sorban levõ lecsengés beállitás változtatásával a hangzás idõtartamát tudjuk befolyásolni.

5. Szólamok választása és alacsony és magas frekvencia hang értékek beállitása

Minden egyedi hang meghatározásához a COMMODORE 64 két különálló POKE utasitást használ - egyet a magas helyiértékhez /MAGAS FREKVENCIA/ és egyet az alacsonyhoz /ALACSONY FREKVENCIA/. Az M függelékben levõ ZENEI HANG ÉRTÉKTÁBLÁZAT megmutatja, hogy a COMMODORE 64 nyolc oktávján belül melyek azok az értékek, amelyeket a POKE utasitásokkal be kell irni a szükséges hang lejátszásához. A magas és alacsony frekvencia értékeit a POKE utasitásokkal az egyes szólamokra különbözõ tárhelyekre kell beirni ez teszi lehetõvé, hogy mind a három szólamot függetlenül programozza 3 szólamu zene vagy különleges hanghatások létrehozásához.

A középsõ /ötödik/ oktáv hangjainak MAGAS ÉS ALACSONY FREKVENCIÁRA POKE-kal beirandó értékei, valamint a POKE utasitásban a különbözõ szólamokra megandandó tárhely cimek találhatók a következõ táblázatban.

SZÓLAM ÉS FREKV.	TÁR-HELY	MINTA HANGJEGYEK – ÖTÖDIK OKTÁV C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
1.SZ/MAG.	54273	34	36	38	40	43	45	48	51	54	57	61	64	68	72
1.SZ/AL.	54272	75	85	126	200	52	198	127	97	111	172	126	188	149	169
2.SZ/MAG.	54280	34	36	38	40	43	45	48	51	54	57	61	64	68	72
2.SZ/AL.	54279	75	85	126	200	52	198	127	97	111	172	126	188	149	169
3.SZ/MAG.	54287	34	36	38	40	43	45	48	51	54	57	61	64	68	72
3.SZ/AL.	54286	75	85	126	200	52	198	127	97	111	172	126	188	149	169

Mint látja, mindegyik szólamra 2 beállitás van, egy MAGAS FREKVENCIA és egy ALACSONY FREKVENCIA beállitás. Ha egy zenei hangot akarunk megszólaltatni, akkor egy értéket kell beirni POKE-kal, a MAGAS FREKVENCIA tárhelyére és egy másikat az ALACSONY FREKVENCIA tárhelyére. A SZÓLAM/FREKVENCIA/HANGÉRTÉK táblázatunkban levõ beállitásokat használva, állitsuk be az 5. oktávbeli C hangot /1. szólam/:

```
POKE 54273,34: POKE 54272,75.
```

Ugyanez a 2. szólamra:

```
POKE 54280,34: POKE 54279,75
```

Programban használva ez igy néz ki /a sorok magyarázatát a program után találja meg/:

```
10 HE=54296:HA=54276:F=54277:MF=54273:AF=54272
20 POKEHE,15:POKEHA,33:POKEF,190
30 FORT=1TO200:POKEMF,34:POKEAF,75:NEXT
40 POKEMF,0:POKEAF,0:POKEHA,0

READY.
```

10: betüknek számokkal való értékadása

20: hangerõ, hullámalak, felfutás/lecsengés beállitása

30: időtartam, magas/alacsony frekvencia beállitás

40: a magas/alacsony frekvencia és hullámalak kitörlése.

EGY DALLAM ELJÁTSZÁSA A COMMODORE 64-N

A következõ program egy dallam szerzésre vagy játszására használható/egy szólamban/ . Két fontos tanulsága van e programnak: elõször figyeljük meg, hogyan rövidítjük a hosszu vezérlõ számokat /tárhely cimeket/ a program elsõ sorában, ettõl kezdve pl. HULLÁMALAK-nak a H-t használhatjuk az 54276-s szám helyett.

A második tanulság a DATA használatára vonatkozik. Ez a program lehetõvé teszi az egyes hangjegyekhez tartozó 3 érték megadását: a hangjegy MAGAS FREKVENCIA értékét, az ALACSONY FREKVENCIA értékét és a hang játszási IDÕTARTAMÁT.

Ennél a dallamnál a 8-ad hanghoz a 125-t, negyed hanghoz 250-t, a pontozott negyedhez 375-t, a félhanghoz 500-t és az egész hanghoz 1000-t adunk meg az időzitõ ciklus határértékeként. Ezek a számértékek növelhetõk vagy csökkenthetõk egy kivánt ütemnek vagy az Ön zenei izlésének megfelelõen.

Annak megvizsgálásához, hogy hogyan lehet egy dallamot megadni, nézze meg a 100-as sort. A "C" hang lejátszásá-

hoz 34-t ill. 75-t adtunk meg a MAGAS ill. ALACSONY FREKVENCIA beállitásaként /ahogy a minta skálából látható volt elõzõleg/ és 250-t adtunk meg a negyed hangra. Igy elõször dalunkban negyed értékü hangot hallunk. A második hang szintén negyed értékü, de ez a D hang ... és igy tovább a dallam végéig. Ön igy meg tud adni majdhogynem bármely dalt annyi további DATA utasitássort hozzátéve, amennyit szükséges. A hangjegy értékek és idõtartam érték megadását folytathatja egyik sorról a másikra, de az egyes sorok elején ki kell irni a DATA kulcsszót. A DATA-1,-1,-1 kell, hogy legyen a program utolsó sora. Ez a sor a dallam végét jelzi.

Irja be a NEW szót az elõzõ program törléséhez és irja be a következõ programot, majd futassa le a RUN paranccsal, hogy meghallja a dal néhány elsõ taktusát.

```
5 HE=54296:HA=54276:FL=54277:MF=54273:AF=54272
8 TE=54278:PF=54275:PA=54274
10 POKEHE,15:POKEHA,65:POKEFL,190:POKEPF,15:POKEPA,15
20 READM
30 READA
40 READT
50 IFM=-1THENEND
60 POKEMF,M:POKEAF,A
70 FORX=T-30TOT-20:POKETE,136:NEXT
80 FORX=1TOT:NEXT:POKEMF,0:POKEAF,0:POKEHA,0
90 GOTO10
100 DATA25,177,200,25,177,100,34,75,300,34,75,250,0,0,50
105 DATA25,177,200,25,177,100,34,75,300,34,75,250,0,0,50
110 DATA38,126,200,38,126,100,43,52,300,43,52,250,0,0,50
115 DATA38,126,200,34,75,100,43,52,300,43,52,500
120 DATA-1,-1,-1
```

HANGHATÁSOK KELTÉSE

A zenével ellentétben hanghatások keltése legtöbbször egyedi, programban megadott tevékenységhez kötõdik, mint pl. ürjátékban a robbanás, melyet az ürrepülõ okoz

egy akadályon keresztül történő áthatolásakor, vagy egy üzleti programban a figyelmeztető hangjelzés, amely jelzi a felhasználónak, hogy hibás tevékenysége folytán éppen le fogja törölni lemezét.

Önnek sok lehetséges választása van, ha különböző hanghatásokat akar kelteni. Itt lO programozási ötletet adunk ahhoz, hogy a hanghatásokkal való kisérletezésben való elindulását elősegitsük:

1. Változtassa a hangerőt mialatt egy hang szól, pl. azért, hogy "visszahang" hatást keltsen!

2. Váltogasson gyorsan két hangot remegési hatás keltésére.

3. Minden szólamra próbáljon különböző hullámalak beállitásokat.

4. Felfutás/lecsengés segitségével módositsa a hangnak a csucsértékére való felfutásának idejét, illetve ezen értékről való csökkenésének sebességét.

5. Tartás/elengedés különböző beállitásaival kiterjesztheljük vagy csökkenthetjük a hanghatás hangerejét vagy hangok sorozatát kombináljuk össze.

6. Több szólamu hatások - egyidőben egynél több hang játszása, az egyes hangokat függetlenül vezérelve pl. úgy, hogy az egyik hang legyen hosszabb, mig a másik rövidebb vagy visszhangként vagy feleletként szolgáljon az első hanghoz.

7. Cserélgessük a skála hangjait, vagy cseréljük az oktávokat a ZENEI HANGTÁBLÁZAT értékeit használva.

8. Használja a négyszög hullámalakot és különböző impulzus beállitásokat /a P függelékben megadott további POKE utasitással/ különféle hatások létrehozására.

9. A zaj hullámalak használatával generáljunk zajhatást szinezett hanghatások hangsulyozására vagy robbanás, lövés, lépések stb. előállitására. Ugyanazok a zenei hangok - amelyekkel dallamot hoztunk létre - felhasználhatók különböző zaj tipusok előállitásához a zaj hullámalakkal.

10. Gyors egymásutánban kombináljon magas/alacsony frekvenciákat különböző oktávokon keresztül

HANGHATÁS MINTÁK KIPRÓBÁLÁSA

A következő programot majdnem minden BASIC programhoz hozzá lehet adni. Azért közöljük, hogy adjunk Önnek néhány programozói ötletet és bemutassuk a COMMODORE 64 által kelthető hanghatások körét.

Figyeljük meg a program rövidítésére alkalmazott fogást a 10. sorban. A hang beállitásokhoz szükséges kényelmetlen, hosszu számokat rövidíteni tudjuk, könnyen használható betükkel /numerikus változókkal/.

A 10-s sor egyszerüen azt jelenti, hogy e könnyen megjegyezhető betük használhatók a hosszu számok helyett. Itt H= hangerő, HA= hullámalak, FL=felfutás/lecsengés, MF= magas frekvencia /1. szólam / és AF= alacsony frekvencia/1. szólam /. Ezután mi ezeket a betüket használjuk programunkban a számok helyett, ... igy programunk rövidebb, gyorsabban tudjuk begépelni és a hangbeállitásokra könnyebben tudunk emlékezni ill. megtalálni.

LÖVÉS HANGJA... 1. SZÓLAM, ZAJ HULLÁMALAK, FOKOZATOSAN CSÖKKENŐ HANGERŐ

```
10 HE=54296:HA=54276:FL=54277:MF=54273:AF=54272
20 FORX=15TO0STEP-1:POKEHE,X:POKEHA,129:POKEFL,15
25 POKEMF,40:POKEAF,200:NEXT
30 POKEHA,0:POKEFL,0
```

8. FEJEZET

HALADÓ ADATKEZELÉS

- READ és DATA utasitások
- Átlagok
- Indexes változók
- Egydimenziós tömbök
- Átlagok revidiálása
- Dimenzió
- Szimulált kockadobás tömbökkel
- Kétdimenziós tömbök

READ /OLVASS/ ÉS DATA /ADAT/ UTASITÁSOK

Korábban láthatta, hogyan lehet közvetlenül a programon belül értékeket rendelni változókhoz /A=2/ és hogyan lehet különböző értékeket hozzárendelni a program futása közben az INPUT utasitással.

Sokszor azonban ezek egyike sem teljesen megfelelő a feladat elvégzésére, amelyet éppen próbál megoldani, különösen ha az sok információt tartalmaz.

Nézzük az alábbi rövid programot:

```
10 READ X
20 PRINT "X MOST! ";X
30 GOTO 10
40 DATA 1,34,10.5,16,234.56
RUN
X MOST!  1
X MOST!  34
X MOST!  10.5
X MOST!  16
X MOST!  234.56

?OUT OF DATA  ERROR IN 10
READY.
```

A 10-s sorban a számitógép egy értéket olvas be a READ utasitással a DATA utasitásból és ezt az értéket hozzárendeli X-hez. Ahányszor ujra a READ utasitás következik a ciklusban, annyiszor a DATA utasitásból a következő érték kerül beolvasásra, és ez az érték lesz az X-hez hozzárendelve, majd a PRINT utasitással kiirva. A számitógépben levő mutató nyomon követi, hogy melyik a következő beolvasásra kerülő érték.

mutató

```
40 DATA 1,34,10.5,16,234.56
```

Amikor az összes értéket felhasználtuk és a gép ujabb READ utasitást végez el, akkor, mivel nincs több érték a READ számára, OUT OF DATA /adat elfogyott/ hibajelzést ir ki.
Fontos, hogy a DATA utasitás formáját pontosan betartsuk:

```
40 DATA 1,34,10.5,16,234.56
```

vesszõ választ el minden tételt

A DATA utasitásban lévõ adatok lehetnek egész, valós /234.65/ vagy lebegõpontos számok. De nem tudunk beolvasni más változókat vagy aritmetikai kifejezéseket a DATA-ban. Ez például helytelen:

```
40 DATA A,23/56,2*5
```

Azonban lehet szöveges változókat használni a READ utasitásban és az ezekhez tartozó információkat a DATA sorban elhelyezni. Elfogadható a következõ:

```
10 FOR X=1 TO 3
15 READ A$
20 PRINT "A$ MOST: ";A$
30 NEXT
40 DATA EZ,AZTAN,VICCES
RUN
A$ MOST: EZ
A$ MOST: AZTAN
A$ MOST: VICCES

READY.
```

Megjegyezzük, hogy itt a READ utasitás egy a *FOR...NEXT* cikluson belül volt elhelyezve. Ez a ciklus annyiszor került végrehajtásra ahány adat volt a DATA utasitásban. Sok esetben a program futása elõtt a DATA utasitásban változtatjuk az értékek számát. Hogy ne kelljen az érték számát megszámolni, de mégse kapjunk "adat elfogyott"

hibajelzést, helyezzünk el a DATA sorban, utolsó értékként egy jelzőt. Ez a jelző olyan érték lehet, amilyen az Ön adatai között nem fordulhat elő, pl. egy negativ, egy nagyon nagy vagy egy nagyon kicsi szám. Amikor ez az érték kerül beolvasásra, akkor a program a következő részre ágazik el.
Ugyanazon adatok ujra felhasználása a programban a RESTORE /állitsd vissza/ utasitással lehetséges, melynek hatására a mutató az adatlista kezdetére mutat. Adjon egy 50-s sor számu sort az előző programhoz:

```
50 GOTO 10
```

Ön mégis kap OUT OF DATA hibajelzést, mivel, ahogy a program a 10-es sorra ugrik vissza adat ujra beolvasáshoz, az adatjelző jelzi, hogy az összes adat fel lett használva. Most adja még hozzá:

```
45 RESTORE
```

és futtassa ujra a programot. A mutató ennek hatására visszaállitódik és az adatokat folyamatosan lehet beolvasni.

ÁTLAGOK

A következő program bemutatja a READ és DATA gyakorlati hasznosságát számhalmazok beolvasásán és ezek átlagának kiszámitásán keresztül.

```
5 AD=0: SZ=0
10 READ X
20 IF X=-1 THEN 50: REM VEGJEL ELLENOR.
25 SZ=SZ+1
30 AD=AD+X: REM VEGOSSZEG NOVELESE
40 GOTO 10
50 PRINT SZ;" ERTEKET OLVASTUNK BE"
60 PRINT "VEGOSSZEG= ";AD
70 PRINT "ATLAG= "; AD/SZ
80 DATA 75,80,62,91,87,93,78,-1
RUN
 7  ERTEKET OLVASTUNK BE
VEGOSSZEG=  566
ATLAG=  80.8571429
```

Az 5-s sorszámu sorban az SZ /számláló/ és a AD /végösszeg/ Ø értékre való beállitása történik. A 10-s sor beolvas egy értéket és ezt hozzárendeli X-hez. A 20-s sor leellenőrzi a jelzőnket/amely itt -1/. Ha a tényleges adatrészből olvas be a READ, akkor az SZ 1-gyel növeszik és X hozzáadódik a végösszeghez.

Amikor a jelző kerül beolvasásra, akkor a program az 50-s sorra ágazik el, ahol a beolvasott értékek száma kerül kiirásra. A 60-s. sor kiirja a végösszeget és a 70-s sorban megkapjuk az átlagot ugy, hogy a végösszeget elosztjuk az értékek számával.
Az adatok végénél alkalmazott jelző lehetővé teszi,hogy tetszőleges számértéket helyezzünk el a DATA utasitásokban - amelyek több soron keresztül irhatók a bemenő értékek számának megszámlálása nélkül.
A READ utasitás egy másfajta alkalmazása az, hogy ugyanabból az adatlistából lehet különböző változókhoz is értékeket rendelni. Ez az információ füzérek és numerikus adatok keveréke is lehet. A következő programban mindezt láthatja ahogy a READ beolvas egy nevet és néhány eredményt /mondjuk tekézésből eredőt/, majd kiirja a nevet, az eredményeket és az eredmények átlagát:

```
10 READ N$,A,B,C
20 PRINT N$;" SOROZATA";A;",";B;",";C;"V
OLT"
30 PRINT "IGY AZ ATLAGA:";(A+B+C)/3
40 PRINT: GOTO 10
50 DATA MIKI,190,185,165,DINI,225,245,19
0
60 DATA JANI,155,185,205,PALI,160,179,18
7
RUN
MIKI SOROZATA 190 , 185 , 165 VOLT
IGY AZ ATLAGA: 180

DINI SOROZATA 225 , 245 , 190 VOLT
IGY AZ ATLAGA: 220
```

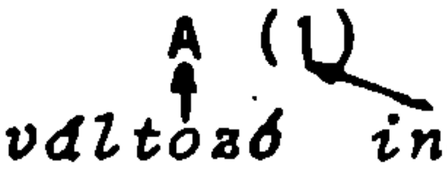
A program futásához a DATA utasitások aszerint vannak felépitve, ahogy a READ utasitás várja az információt: egy név /egy füzér/, azután három érték. Más szóval az N$

először a DATA-ból a "MIKI"-t kapja, A-hoz a READ az adatok közül a 190-t rendeli, míg B-hez a 185-t és C-hez a 165-t. A folyamat azután ugyanígy ismétlődik az információk további részére. /Dini és eredményei,Jani és eredményei, valamint Pali és eredményei./

INDEXES VÁLTOZÓK

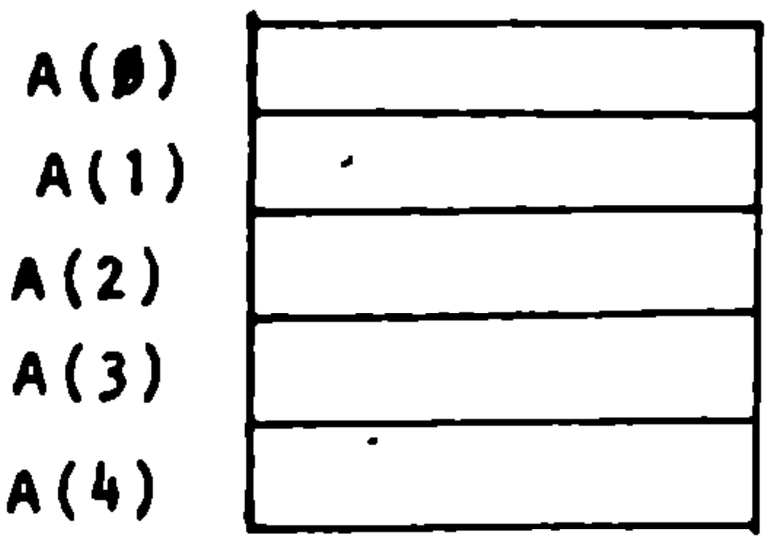
Eddig az értékek ábrázolására csak egyszerű BASIC változókat használtunk, ugymint A,A$, és SZ. Ezek egyetlen betűből álltak, melyet esetleg egy betű vagy egy számjegy követett. Nem valószinü, hogy olyan programot kellene irnia, amelyben több változó névre van szükség, mint amennyi lehetséges a rendelkezésre álló számok és betük kombinációja alapján. De korlátozottak a lehetőségek a változóknak a programokban való használati módja tekintetében.

Most vezessük be az indexes változók fogalmát.



Ezt ugy mondjuk, hogy "a egy". Az indexes változó egy változónévből áll, amelyet egy kerek zárójelbe zárt index követ. Figyelje meg a különbséget az A,A1 és A(1) között. Mindegyik a másiktól független, önálló változó. Csak az A (1) indexes változó.

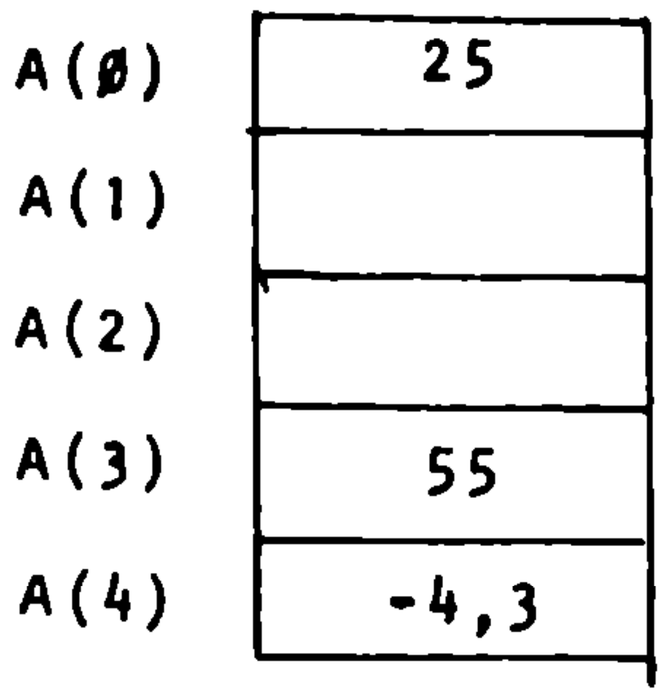
Az indexes változók ugyanúgy, mint az egyszerü változók, a számitógépen belüli tárhelyek neveként értelmezhetők. Az indexes változókat mint információ tároló rekeszeket képzeljük el, éppen ugy mint az egyszerü változókat:



Ha azt irja, hogy:

```
10 A(Ø) = 25 : A(3) = 55 : A(4)= - 45.3
```

akkor a tár a következőképpen néz ki:



Az indexes változóknak ezt a halmazát tömbnek nevezzük, ugyanúgy. Esetünk egy egy-dimenziós tömb. Később több-dimenziós tömböket is bevezetünk.

Az indexek bonyolultabb változók vagy müveletek is lehetnek.

Az alábbiak megengedett indexes változók:

```
A(X)   A(X+1)   A(2+1)   A(1*3)
```

A zárójelben levő kifejezések a 2. fejezetben leirt aritmetikai müveletekre vonatkozó szabályok szerint kerülnek kiértékelésre.

Most az alapszabályok ismerete után nézzük meg, hogyan lehet használni az indexes változókat! Az egyik lehetőség az INPUT vagy READ utasitásokkal beolvasott adatok tárolása.

Most számitsunk átlagot más módon, az indexes változók használatával.

Ezt a programot egyszerübben is el lehetett volna készi-teni, de igy bemutatja az indexes változók müködését. A 10. sor a bemenő adatok számát kérdezi meg. Ez a változó, X, a ciklusszámláló szerepét tölti be abban a ciklusban, amelyikben a bemenő értékek beirásra és a B indexes változóhoz hozzárendelésre kerülnek.

```
5 PRINT CHR$(147)
10 INPUT "HANY SZAM LESZ";X
20 FOR A=1 TO X
30 PRINT "ADD MEG A";A;". SZAMOT: ";:INPUT B(A)
40 NEXT .
50 SU=0
60 FOR A=1 TO X
70 SU=SU+B(A)
80 NEXT
90 PRINT: PRINT "ATLAG =";SU/X

RUN

HANY SZAM LESZ? 5
ADD MEG A 1 . SZAMOT: ? 125
ADD MEG A 2 . SZAMOT: ? 167
ADD MEG A 3 . SZAMOT: ? 189
ADD MEG A 4 . SZAMOT: ? 167
ADD MEG A 5 . SZAMOT: ? 158

ATLAG = 161.2
```

A beolvasó ciklus mindenegyes végrehajtása során A 1-gyel növekszik és igy a következő bejövő értéket az A tömb következő eleméhez rendeljük hozzá. Elõször például a ciklusban A=1 és igy az első bejövő érték a B(1)-be kerül. A következő lépésben A=2 lesz, ekkor az érték a B(2)-be kerül és ez igy megy mindaddig mig a bejövő értékek el nem fogynak.

Most jön egy nagy különbség. Ha minden értéket beadtunk, akkor a tömbben tárolva vannak,készen arra,hogy valamilyen módon használjuk azokat. Eddig INPUT vagy READ ciklus használatánál tudtunk ugyan egy összegzést végezni a ciklus minden egyes futtatásával, de nem tudtuk az információk ujraolvasása nélkül visszakapni az egyes értékeket.

Az 50-es sorszámu sortól 80-ig tartó ujabb ciklusban a tömb különböző elemeit összeadjuk, majd ezután kiirjuk az átlagot. Ez a különálló programrész mutatja, hogy minden érték tárolt és szükség esetén bármikor elérhető.

Annak bizonyitására, hogy az egyes értékek valóban elkü-

lönülten kerültek tárolásra a tömbben, közvetlenül az elõzõ program végrehajtása után gépelje be a következõket:

```
FOR A=1 TO 5: ?B(A),: NEXT
 123          167          189          167
 150
READY.
```

A kiiratás az éppen Ön által megadott értékeket mutatja, mint a tömb tartalmát.

DIMENZIÓ

Ha 10 számnál több adatot próbált bevinni az elõzõ példában akkor "dimenzió hiba" /DIMENSION ERROR/ hibajelzést kapott. Ahol olyan tömbre van szükség, amelynek legfeljebb 11 eleme van /az index egy-dimenziós tömbre 0-tól 10-ig tart/, ugyanugy lehet eljárni mint az egyszerü változókkal, bárhol lehet használni azokat a programon belül. 11-nél több elemü tömböket "deklarálni" kell egy dimenzió utasitással

A programot bõvitsük a következõ sorral:

```
5 DIM B(100)
```

Ez megadja a számitógépnek, hogy maximum 101 eleme lehet a tömbnek.

Dimenzió utasitásban van lehetõség változó használatára, pl. a következõ sor helyettesitheti az 5-s sort /nem szabad elfelejteni az 5. sor kitörlését/:

```
15 DIM B(X)
```

Ez úgy ad dimenziót a tömbnek, hogy pontosan ki legyen töltve a beirandó értékekkel.

Legyen óvatos! Ha egyszer már egy tömb dimenzióját megadta, akkor nem lehet ujra dimenziót adni neki a program egy másik részében.
Ha összetett tömbjei vannak, akkor mindegyiknek ugyanabban a sorban adhat dimenziót például igy:

```
10 DIM C(20), D(50), E(40)
```

KOCKADOBÁS SZIMULÁLÁSA TÖMBÖKKEL

Ahogy a programok egyre bonyolultabbak, az indexes változók használatával csökkenteni lehet a szükséges utasitások számát és igy egyszerübben lehet megirni a programot.

Egyszerü indexes változót használhatunk például kockadobások eredményeinek nyomon követéséhez:

```
1 REM DOBOKOCKA SZIMULACIO: PRINT CHR$(147)
10 INPUT "HANY DOBAS LEGYEN";X
20 FOR D=1 TO X
30 H=INT(6*RND(1))+1
40 F(H) = F(H)+1
50 NEXT D
60 PRINT "OLDAL","HANYSZOR DOBVA"
70 FOR OL = 1 TO 6: PRINT OL, F(OL): NEXT
```

Az oldalakra létrehozunk egy F tömböt, amelyben nyomon tudjuk követni, hogy hányszor mutat a kocka egy bizonyos oldalt.
Például, ahányszor 2-őt mutat, F(2) értéke 1-el nagyobb lesz. Azzal, hogy a tömb ugyanazon elemét használjuk a dobott oldal aktuális értékének tárolására, kiküszöbölünk 5 további változót (mindenegyes oldalra egyet) és számos utasitást a dobások eredményének ellenőrzéséhez és megjelenitéséhez.
A 10-s sor arra kérdez rá, hogy hány dobás legyen szimulálva.

A 20-as sor létrehozza a ciklust, amely végrehajtja a véletlenszerü dobásokat és növeli az egyes dobások után

a tömb megfelelő elemét.

A megadott számu dobás elvégzése után a 60-as sor kiirja a fejlécet és a 70-s sor kiirja, hogy az egyes oldalakat hányszor mutatta a kocka.

Egy egyszerü futás ilyen, mint az alábbi:

```
RUN
HANY DOBAS LEGYEN? 1000
OLDAL      HANYSZOR DOBVA
1            174
2            168
3            179
4            156
5            162
6            161
```

Ahogy elnézzük, a kocka nem volt hamisitva!

Összehasonlitásként megmutatjuk ugyanezt a programot, de most indexes változók használata nélkül.
Ne gépeljük le, de vegyük észre a szükséges többlet utasitásokat.

```
10 INPUT "HANY DOBAS LEGYEN";X
20 FOR D=1 TO X
30 H=INT(6*RND(1))+1
40 IF H = 1 THEN F1 = F1 + 1
41 IF H = 2 THEN F2 = F2 + 1
42 IF H = 3 THEN F3 = F3 + 1
43 IF H = 4 THEN F4 = F4 + 1
44 IF H = 5 THEN F5 = F5 + 1
45 IF H = 6 THEN F6 = F6 + 1
50 NEXT D
60 PRINT "OLDAL","HANYSZOR DOBVA"
70 PRINT "1", F1
71 PRINT "2", F2
72 PRINT "3", F3
73 PRINT "4", F4
74 PRINT "5", F5
75 PRINT "6", F6
80 END
```

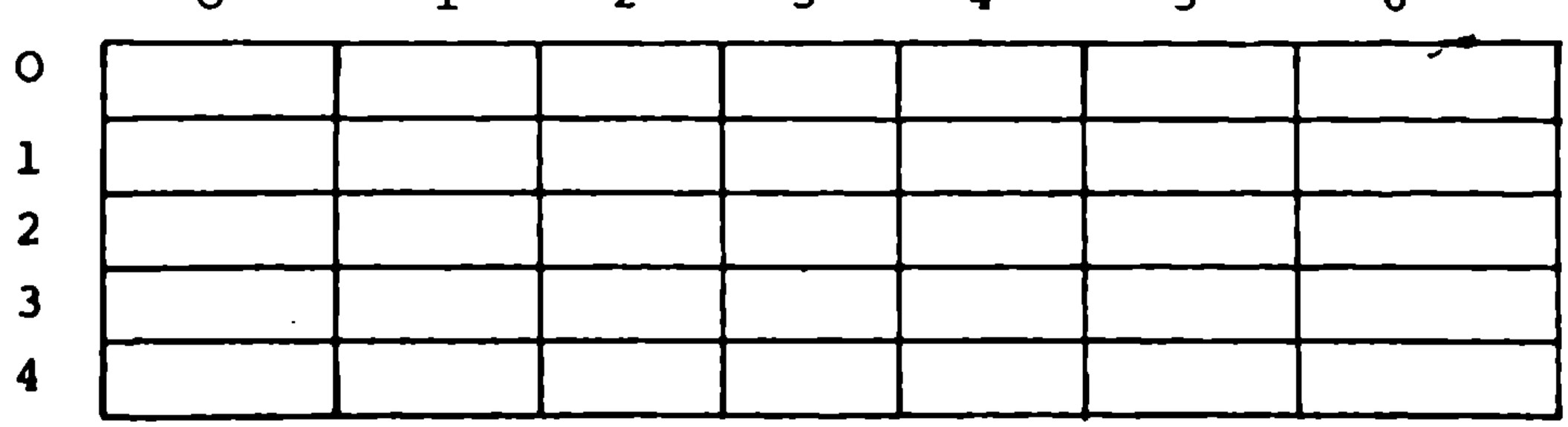
A program 8-ról 17 sorra növekedett. Nagyobb programban az indexes változók használatából eredő hely nyereség még sokkal kiáltóbb lesz.

KÉTDIMENZIÓS TÖMBÖK

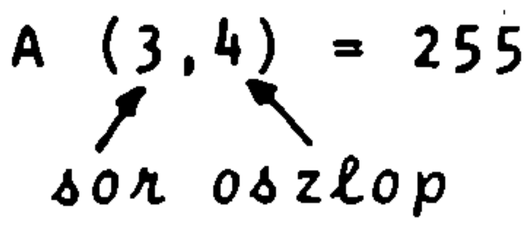
Ebben a fejezetben korábban egy-dimenziós tömbbel kísérletezett.
Ezt a tömbtipust ugy szemlélhettük, mint egymásra rakott táron belüli dobozok halmaza, amelyek mindegyike egy-egy tömb elemet tartalmaz. Mit gondol, hogy néz ki egy két dimenziós tömb?

Elõszöris, a kétdimenziós tömböt a következõképpen irjuk le:

és a memóriában kétdimenziós rácsként reprezentálhatjuk.



Az indexeket úgy képzelhetjük el, hogy a táblázat azon sorának és oszlopának az értékét adják, ahol a tömb adott eleme tárolva van.



	0	1	2	3	4	5	6
0							
1							
2							
3					255		
4							

Ha a 255-s értéket az A(3,4)-hez rendeljük, akkor ugy képzelhetjük, hogy a 255 a táblázat 3. sorának 4. oszlopában helyezkedik el.

Két dimenziós tömbökre ugyanazok a szabályok érvényesek, mint az egy dimenziósokra:

Dimenziót kell adni nekik:

```
DIM A(20,20)
```

Adat hozzárendelés:

```
A(1,1) = 255
```

Tömb értékek más változókhoz való hozzárendelése:

```
AB = A(1,1)
```

Érték kiiratás PRINT-tel:

```
PRINT A(1,1)
```

Ha a két dimenziós tömbök úgy müködnek, mint a kisebb változataik, akkor milyen további lehetőségeket nyujtanak a kiterjesztett tömbök?

Próbálja a következőt: vajon milyen módon használhat fel egy két-dimenziós tömböt egy klub-kérdőiv eredményeinek kiértékelésére, amely 4 kérdésből áll és az egyes kérdésekre 3 lehetséges válasz adható? A kérdőiv pl. ilyen:

Klub kérdőiv

1. Kérdés: Előnyben részesiti-e az 1. számu döntést?

☐ 1-IGEN ☐ 2-NEM ☐ 3-NEM TUDOM stb...

Erre a problémára a tömbtáblázat a következőképpen ábrázolható:

	VÁLASZOK		
	IGEN	NEM	HATÁROZATLAN
1. Kérdés			
2. Kérdés			
3. Kérdés			
4. Kérdés			

A tényleges kérdõiv értékelést létrehozó program egy lehetséges változata látható a következõ oldalon.

Ez a program sok olyan programozói leleményt tartalmaz, amelyet már bemutattunk. Ha nincs is éppen szüksége most e programra, nézze meg, tudja-e követni, hogyan müködik.

A program lelke egy 4x3-s két-dimenziós tömb, A (4,3). Minden egyes kérdésre adott egyes lehetséges válaszok összesitése a tömb megfelelõ elemében van elhelyezve.

Az egyszerüség kedvéért nem használjuk az elsõ sort és az elsõ oszlopot A(O,O)-tól A (O,4)-ig ill. A (1,O)-tól A (4,O)-ig . Emlékezzen azonban arra, hogy ezek az elemek bármely Ön által tervezett tömbben mindig jelen vannak.

Gyakorlatban, ha az 1. kérdésekre a válasz IGEN, akkor A 1,1 -t növeljük 1-el az 1. az 1. kérdéshez és az 1. oszlop egy IGEN válaszhoz. Egy NEM válasz a 3. kérdésre egy hozzáadást eredményezi az A 3,2 elemhez és igy tovább.

```
20 PRINT "□"
25 KI=KI+1: REM VALASZOLOK SZAMA
30 FOR K = 1 TO 4
40 PRINT K;". KERDES"
50 PRINT "1-IGEN  2-NEM  3-HATAROZATLAN"
60 PRINT "MI VOLT A VALASZ?: ";
61 GET V: IF V<1 OR V>3 THEN 61
65 PRINT V: PRINT
70 E(K,V)=E(K,V) + 1: REM ELEMEK NOVELESE
80 NEXT K
85 PRINT
90 PRINT "VAN TOVABBI KITOLTOTT KERDOIV(I/N)?"
100 GET V$: IF V$="" THEN 100
110 IF V$ = "I" THEN 20
120 IF V$ <> "N" THEN 100
130 PRINT "□";"A VALASZOLOK SZAMA:";KI:PRINT
140 PRINT SPC(18);"VALASZ"
141 PRINT "KERDES","IGEN","NEM","NEM TUDJA"
142 PRINT "------","----------------------------"
150 FOR K = 1 TO 4
160 PRINT K,E(K,1),E(K,2),E(K,3)
170 NEXT K
180 END
```

A programot futtatva, és a négy kérdésre tizszer - különböző módon - válaszolva ilyen eredményt kaphatunk:

```
A VALASZOLOK SZAMA: 10

                    VALASZ
KERDES      IGEN        NEM          NEM TUDJA
------      ----------------------------------
 1           4           5            1
 2           3           6            1
 3           2           4            4
 4           6           1            3
```

FÜGGELÉKEK

BEVEZETÉS

Most, hogy Ön bizalmas kapcsolatba került a COMMODORE 64-el, tudatni akarjuk Önnel, hogy segitségünk itt nem ért véget. Talán nem tudja, de a Commodore 23 éves multra tekint vissza. 1970-ben vezette be az első önálló személyi számitógépet /PET/. Azóta a világ számos országában a vezető számitógépes társaság lett. Képességünk a saját számitógép integrált áramköreink tervezésére és gyártására ujabb és jobb személyi számitógép előállitást teszi lehetővé az Ön részére, alacsonyabb áron, mint amire Ön számitana e kimagasló technikai szint mellett.

A COMMODORE elkötelezte magát nemcsak az Ön, a végső felhasználó támogatására, de a kereskedőére is akitől Ön a számitógépet vásárolta, a folyóiratokéra amelyek népszerüsitik azokat a cikkeket, amelyek uj alkalmazásokat és technikát mutatnak be és ez is fontos - a szoftver fejlesztőkére, akik előállitják az Ön számitógépén használandó programokat modulokban, lemezen vagy kazettán. Mi biztatjuk Önt, hogy hozzon létre, vagy csatlakozzon már létező Commodore felhasználói klubhoz, ahol Ön tanulhat uj technikákat, kicserélheti ötleteit és a felfedezéseit is megoszthatja. Mi két különálló magazint adunk ki, amelyek programtippeket tartalmaznak, uj termékekről szolgáltatnak információkat és a számitógép alkalmazására adnak ötleteket.

A következő függelékek táblázatokat és egyéb információkat tartalmaznak, amelyek segitik Önt abban, hogy programjait gyorsabban és hatékonyabban irja meg Commodore 64 gépére. Tartalmaznak továbbá a Commodore termékek széles köréről is fontos információt, amely Önt érdekelheti.

- . -

COMMODORE 64 TARTOZÉKAI ÉS SZOFTVERE

TARTOZÉKOK

A COMMODORE 64 támogatja a COMMODORE VIC 20 tároló eszközeit és tartozékait - a C2N kazettás egységet, modemet, nyomtatót - igy az Ön rendszere bővithető, hogy lépést tartson a változtatási szükségletekkel.

- C2N kazettás egység. Ez az olcsó szalagos egység képes programokat és adatokat tárolni kazettás szalagon és később visszajátszani ezeket. Előre megirt programok betöltésére is lehet használni.

- Lemez - Az egyes lemezegység szabványos 5 1/4 hüvelykes hajlékony lemezt használ, amelyeknek mérete valamivel kisebb a 45 fordulatu hanglemezeknél, és amelyeken programot és adatot lehet tárolni. A lemezek gyorsabb adathozzáférést biztositanak és közel 170 000 karakternyi információt tárolnak egyenként. A lemez egységek "intelligensek", saját mikroprocesszorjuk és tárjuk van. A lemezek nem igényelnek erőforrást a COMMODORE 64-től, mint pl. a főtár egy részének használatát.

- Modem - egy olcsó távközlési eszköz, a VICMODEM lehetővé teszi a más számitógépekhez való hozzáférést, szokványos telefon vonalakon. A felhasználók igy hozzáférhetnek nagy adatbázisok teljes forrásához.

- Nyomtató - a nyomtatók programok, adatok vagy grafikák másolatainak kiiratását teszi lehetővé. Különböző tipusu nyomtatók állnak rendelkezésre, különböző szolgáltatásokkal és árakon. A 30 karakter /másodperc sebességü VIC pont mátrix nyomtató egyszerü papirt és egyéb olcsó tartozékokat használ. A nyomatók közvetlenül kapcsolódnak a Commodore 64-hez, minden

egyéb interfész nélkül.

- Interfész modulok - számos különleges modul teszi lehetővé a Commodore 64 részére a különböző szabványos egységek, ugymint modemek, nyomtatók, vezérlők, műszerek - a rendszerhez való kapcsolódását.

A speciális IEEE-488-as modullal a Commodore 64 támogatni fogja a CBM perifériák teljes tartományát, beleértve a lemezegységeket és printereket is.

Továbbá a Z80 modul lehetővé teszi, hogy a Commodore 64-n a CP/M operációs rendszer fusson, megadva ezzel a legnagyobb mikroszámitógép alkalmazási bázis hozzáférésének lehetőségét.

SZOFTVER

A szoftver különböző kategóriáit kinálják a COMMODORE részére, amelyek ellátják Önt különböző személyes, szórakozási és oktatási alkalmazási lehetőségek széles választékával

ÜZLETI SEGÉDLETEK

- Az "Elektronikus Könyvelő iv" csomag lehetővé teszi a költségvetési tervezéseket és a "mi történik, ha..." elemzések elvégzését. És az opcionális grafikus programmal létre lehet hozni a kérdőiv adataiból jellemző grafikonokat.

- Pénzügyi tervezések, ugymint kölcsönzések amortizációja, könnyen elvégezhető a Pénzügyi Tervezés Csomag segitségével.

- Számos Hivatásos Időbeosztó program segitik a megbeszélések és a munka beosztását.

- Könnyen használható Adatbázis programok lehetővé teszik az információk, pl. cimlisták, telefon listák, leltárak stb.nyomon követését, és az információkat használható formába szervezik.

- A Hivatásos Szövegfeldolgozó programok a Commodore 64-et teljes képességü szövegfeldolgozóvá alakitják.

Begépelni és átnézni jegyzeteket, leveleket és más szöveganyagokat gyerekjáték lesz.

SZÓRAKOZTATÁS

- A Commodore 64-hez csatlakoztatható modulokon a magasabb minőségü játékokhoz lehet hozzájutni, élvezetes órákat szerezve magunknak. Ezek a programok felhasználják a finom felbontásu grafikát és a Commodore 64 adta teljes hanglehetőséget.

- Gépe lehetővé teszi Önnek az összes MAX-on meglévő érdekes és izgalmas játék lehetőséget, mert ez a két gép teljesen kompatibilis modulokat használ.

OKTATÁS

- A COMMODORE 64 egy olyan oktató, amely sohasem fárad el és mindig odafigyel. A sokféle PET oktató programok legtöbbje elérhető, ezenkivül a majd elérhető további oktató nyelvek lehetővé fogják tenni a Commodore 64 részére a PILOT, LOGO és egyéb kulcsfontosságu csomagok használatát.

KAZETTA FEJLETT HASZNÁLATA

Amellett, hogy az Ön programjainak másolatait lehet õrizni a szalagon, a gép tárolni tud változó értéke-
ket és egyéb adat tételeket adatállománynak v. file-1/
nak nevezett csoportositásban.

Ez még több információ tárolását teszi lehetõvé, mint amennyit a számitógép fõ tárjában tartani tud egyidõ-
ben.

Adatállományokkal kapcsolatosak az OPEN, CLOSE, PRINT#, INPUT és GET# utasitások. Az ST rendszerváltozó el-
lenõrzi a szalag jelzéseket.

Az adatok szalagra való irásában ugyanazokat az elve-
ket követjük, mint a számitógép képernyõjére való in-
formáció irásnál. De az információnak a képernyõre PRINT utasitással való kiirása helyett az információt a szalagra irjuk a PRINT utasitás PRINT# változatát használva.

A következõ program illusztrálja, hogyan müködik ez:

```
10 PRINT "IRAS-A-KAZETTARA PROGRAM"
20 OPEN 1,1,1,"ADATALLOMANY"
30 PRINT "IRD BE A TAROLANDO ADATOT"
40 PRINT "VAGY AZT, HOGY: VEGE"
50 PRINT
60 INPUT "ADAT";A$
70 PRINT#1,A$
80 IF A$ <> "VEGE" THEN 50
90 PRINT
100 PRINT "ALLOMANY LEZARVA"
110 CLOSE 1
```

1/ ejtsd: fájl

Az elsõ dolog amit tenni kell, hogy megnyitunk egy adatállományt /esetünkben az ADATALLOMANY-t/ az OPEN /nyit/ utasitással. A 2Ø sorban ez megtörténik.

A 6Ø. sorban jelzi, hogy kéri a tárolandó adatot. A 7Ø. sor irja a szalagra, amit Ön begépelt - és ami az A$ változóban van. Az eljárás igy folytatódik.

Ha "VEGE"-t gépel be,akkor a 11Ø. sor az adatállományt lezárja a CLOSE /zárj/ utasitással.

Az információ visszaszerzéséhez, tekercseljük vissza a szalagot, és próbáljuk ezt:

```
10 PRINT "KAZETTA-OLVASO PROGRAM"
20 OPEN 1,1,0,"ADATALLOMANY"
30 PRINT "ALLOMANY NYITVA"
40 PRINT
50 INPUT#1,A$
60 PRINT A$
70 IF A$ = "VEGE" THEN END
80 GOTO 40
```

Ujra az "ADATÁLLOMÁNY" nevü adatállományt kell elõször megnyitni. Az 5Ø. sorban a program a szalagról várja az A$ -t és a 6Ø. sorban a képernyõre kiirja azt. Azután amig "VEGE"-t nem talál, ismétlõdik az eljárás, ekkor azonban vége a programnak.

A GET változata a GET#, szintén adatok szalagról való visszaolvasásához használható. Az 5Ø.-8Ø. sorokat irjuk át a programban:

```
50 GET#1,A$
60 IF A$ = "" THEN END
70 PRINT A$,ASC(A$)
80 GOTO50
```

COMMODORE 64 BASIC

Ez a kézikönyv bevezette Önt a BASIC nyelv ismeretébe és az alkalmazott szókészlet egy részébe, - elegendően ahhoz, hogy ráérezzen a számitógép programozásra. Ez a függelék a Commodore 64 szabályainak /SZINTAXIS/ teljes listáját adja a rövid leirásokkal együtt. Kérjük próbálja ki ezeket az utasitásokat. Emlékezzen arra, hogy a számitógép a programok begépelése révén nem tud maradandó kárt okozni, tehát a legjobb módszer a számitógép megismeréséhez a gyakorlat.

Ez a függelék a BASIC különböző tipusu müveletei szerint van csoportositva, a következő módon:

1. Változók és müveletek:

   különböző tipusu változók, megengedett változó nevek, valamint aritmetikai és logikai müveletek leirása.

2. Parancsok:

   a programokat kezelő, szerkesztő, tároló és törlő parancsok leirása.

3. Utasitások:

   a programok számozott soraikon használt BASIC program utasitások leirása.

4. Függvények:

   füzér, numerikus és kiiró függvények leirása.

VÁLTOZÓK

A Commodore 64 a BASIC-ben háromféle változót használ. Ezek a valós szám, az egész szám és a füzér /alfanumeri-

kus/ tipusu változók.

A változó nevek egy betübõl, egy betübõl és a betü után következõ számból vagy pedig két betübõl állnak.

Egy egész tipusu változó a változó neve után irt százalék /%/ jellel van megkülönböztetve. A füzér tipusu változók neve után egy dollár /$/ jelet kell tenni.

Példák:

Valós változók: A, A5, BZ

Egész tipusu változók: A%, A5%, BZ%

Füzér tipusu változók: A$, A5$, BZ$

A tömbök azonos nevü változók sorozatából állnak, a tömb elemeit megkülönböztetõ számokkal lehet jellemezni. A tömböket a DIM utasitással határozzuk meg, és lehetne valós, egész vagy füzér tipusu változók. A tömbváltozó nevét kerek zárójelbe irt szám követi a listán.

A(7), BZ(11), A$(50), PT(20,20)

Megjegyzés:

Van háromféle változó, amelyik a COMMODORE 64 részére van lefoglalva, és amelyeket nem lehet másképpen definiálni. Ezek az ST, TI, és TI$.

ST a bemeneti/kimeneti /INPUT/OUTPUT/ müveletekhez kapcsolódó állapot változó. ST értéke változik, ha egy lemezrõl vagy egy szalagról történõ programbetöltéssel probléma van.

TI és TI$ olyan változók, amelyek a Commodore 64-be épitett valós-idõ órával vannak rokonságban. A TI változó a másodperc minden hatvanad részénél ugrik egyet. A számitógép bekapcsolásakor 0-ról indul, és csak TI$ értékének változtatásakor cserélõdik.

TI$ olyan füzér, amelyet a rendszer állandóan változtat. Az elsõ két karakter az órák számát tartal-

mazza, a 3. és 4. karakter a perceket, az 5. és 6. karakter pedig a másodperceket. Ennek a változónak adhatunk tetszőleges számértéket, amelyről tovább fog haladni.

TI$ = "101530" beállitja az órát 10 óra 15 perc 30 másodpercre.

Ez az óra a gép kikapcsolásakor törlődik, és a rendszer bekapcsolásakor 0-tól indul.

MÜVELETEK

Az aritmetikai müveletek a következők:

+	összeadás
-	kivonás
*	szorzás
/	osztás
	hatványozás

Egy olyan sorban, amely egynél több müveletet tartalmaz, egy adott sorrend szerint történnek minden esetben a müveletek. Ha ugyanabban a sorban különböző müveleteket használunk, akkor a számitógép az elvégzési sorrendet a következőképpen határozza meg: első a hatványozás. A következő a szorzás és osztás és végül az összeadás és kivonás.

A müveletek sorrendjét zárójellel meg lehet változtatni. Először a zárójelben levő müveletet végzi el a többi előtt.

Müveletek vannak egyenlőségekre és egyenlőtlenségekre is:

=	egyenlő
<	kisebb, mint
>	nagyobb, mint

< = kisebb, mint vagy egyenlő

> = nagyobb, mint vagy egyenlő

< > nem egyenlő

Végül van még három logikai művelet:

AND ÉS

OR VAGY

NOT NEGÁCIÓ

Ezeket leggyakrabban összehasonlitások összekapcsolására használjuk az IF ... THEN utasitásnál.

Például:

```
IF A = B AND C = D THEN 100   /mindkettőnek igaznak kell
                              lenni/

IF A = B OR C = D THEN 100    /legalább az egyiknek kell
                              igaznak lenni/
```

PARANCSOK

CONT /CONTINUE, folytasd/

Ezt az utasitást a megszakitott program ujrainditására használjuk, ha a STOP billentyü,a STOP utasitás vagy az END utasitás alkalmazása következtében állt le a program.
A program onnan indul ujra, ahonnan kilépett.

A CONT parancs nem müködik, ha a programon változtatak vagy hozzáadtak uj sorokat, vagy ha a program hibával állt le, illetve ha Ön az ujrainditás kisérlete előtt hibát okozott. Ebben az esetben kapjuk a következő jelzést: CAN'T CONTINUE ERROR.

LIST /listáz/

A LIST parancs lehetővé teszi, hogy a memoriában levő

BASIC program sorait megnézze. Le lehet kérdezni és kiiratni a teljes programot vagy csak egyes sorokat.

LIST	a teljes programot irja ki
LIST 10-	csak a 10. sortól kezdődően végig
LIST 10	csak a 10. sort
LIST -10	az elejétől irja ki a 10. sorig
LIST 10-20	a 10. sortól irja ki a 20. sorig

LOAD /tölt/

Ez a parancs a programot átviszi szalagról vagy lemezről a tárba ugy, hogy a program használható lesz. Ha beirja a LOAD-t és a *RETURN*-t leüti, akkor az első programot megkeresi a kazettán, és elhelyezi a tárban. A parancsot követheti az idézőjelek közé tett programnév. A nevet követheti egy vessző és egy szám vagy egy numerikus változó, amelyik azt a készülék számot adja meg, ahonnan a program jön.

Ha nem adunk meg készülékszámot, akkor a COMMODORE 64 az 1 számu készüléket tételezi fel, amelyik a kazettás egység. A másik készülék, amit a LOAD paranccsal általában hivunk, a lemez meghajtó egység, amelynek a száma a 8.

LOAD	beolvassa következő programot a szalagról
LOAD "HELLO"	megkeresi a szalagon a HELLO nevü programot, és ha megtalálta, betölti
LOAD A$	keresi azt a programot, amelynek neve az A$ változóban van

LOAD "HELLO",8	megkeresi a lemez egységen a "HELLO" nevü programot
LOAD "*", 8	betölti az első programot a lemezről.

Másodlagos cimként meg kell szabni az 1-t ha a gépi kodu programot kiván betölteni anélkül, hogy a tárban áthelyezésre kerülne.

LOAD "M/C PROGRAM",1,1	áthelyezés nélkül betölti szalagról a gépi kodu programot.

NEW /uj/

Ez a parancs törli a tárban lévő teljes programot és azokat a változókat, amelyek használatba kerülhettek. Hacsak a programot nem mentette ki SAVE-vel, elvész.

ÓVATOS LEGYEN E PARANCS HASZNÁLATÁNÁL!

A NEW parancsot úgy is lehet használni, mint egy BASIC program utasitást. Amikor a program eléri azt a sort, a programot törli. Ez hasznos, ha a program elvégzése után mindent "tiszta" állapotba akar tenni.

RUN /fuss/

Ez a parancs okozza a program végrehajtását, ha már a program a tárba van töltve. Ha a RUN-t követően nincs sorszám, akkor a számitógép a legalacsonyabb sorszámtól kezdődően indit. Ha sorszám adott, akkor a program a megadott sorszámtól kezdve hajtódik végre.

RUN	a legalacsonyabb sorszámu sortól indit
RUN 100	a 100. sor végrehajtásáról indul
RUN X	UNDEFINED STATEMENT ERROR /határozatlan utasitás hiba/. Mindig egy határozott sorszámot kell megadni, nem változót.

SAVE /ments/

Ez a parancs tárolni fogja a tárban éppen levő programot kazettára vagy lemezre. Ha begépeli a SAVE-t és a *RETURN*-t is leüti, a program ki lesz mentve a kazettára. A számitógép nem tudhatja, hogy a szalagon volt e már program, igy ügyeljen a szalagokra, mert értékes programjait igy letörölheti.

Ha a SAVE-t egy utána következő idézőjelbe tett névvel vagy egy füzér változóval gépeli be, akkor a programnak a számitógép ezt a nevet adja, igy az könynyebben megtalálható a későbbiekben. A nevet követheti itt is a készülék száma.

A készülékszám után egy vesszõ és egy második szám következhet, amely lehet O vagy l. Ha ez a második szám l, akkor a Commodore 64 a program után szalagvég /END-OF-TAPE/ jelet tesz. Ez jelzi a számitógépnek, hogy ne keressen tovább a szalagon, ha netán további LOAD parancsot adna.Ha Ön megpróbál betölteni LOAD-dal programot, és a számitógép ilyen jelet talál FILE NOT FOUND ERROR /file-t nem talált hiba/ jelzést ad.

SAVE	kimenti a programot név nélkül a szalagra
SAVE "HELLO"	"HELLO" névvel viszi ki a szalagra a programot
SAVE A$	az A$-ban levõ névvel irja ki a programot a szalagra
SAVE "HELLO",8	"HELLO" névvel tárolja a programot a lemezre
SAVE "HELLO",1,1	"HELLO" névvel tárolja a programot szalagon és a program után szalagvég jelet ir fel.

VERIFY /ellenõriz/

Ennek a parancsnak a hatására a számitógép leellenõrzi, hogy a lemezen vagy szalagon lévõ program megegyezik e

a memóriában lévővel. Ez bizonyitéka annak, hogy valóban sikerült kimenteni a programot, a szalag vagy lemez nem rossz, illetve a kimentés alatt nem volt meghibásodás. A VERIFY minden nélkül a parancs után a Conmodore 64-et arra birja, hogy hasonlitsa össze a szalagon levő soron következő programot, névre való tekintet nélkül, a tárban lévővel.

Ha a VERIFY-t egy programnév vagy füzér változó követi, akkor kikeresi a megfelelő programot és azután ellenőrzi azt. Készülék számokat is tartalmazhat a VERIFY parancs.

VERIFY	a szalagon levő következő programot ellenőrzi
VERIFY "HELLO"	kikeresi a "HELLO"-t és összehasonlitja a tárban lévővel
VERIFY "HELLO",8	kikeresi a "HELLO"-t a lemezen és azután ellenőrzi.

Annak ellenőrzésére, hogy egy program már a szalagon van irja be azt, hogy VERIFY, a számitógép megmondja, melyik programot találta meg /ha van egyáltalán/.

UTASITÁSOK

CLOSE /zár/

Ez a parancs teszi teljessé és zárja le az OPEN által megnyitott adatállományokat. A CLOSE-t követő szám annak az adatállománynak a száma, amelyet le akarunk zárni.

CLOSE 2 csak a 2-es adatállományt zárja le.

CLR /CLEAR, tisztit/

Ez a parancs törli a tárban levő összes változót, de sértetlenül hagyja magát a programot. Ez a parancs automatikusan végrehajtódik, amikor egy RUN parancsot

adunk ki.

CMD /COMMAND, parancs/

A CMD azt a kimenetet, amely szokásosan a képernyõre kerülne /ugy mint a PRINT, LIST utasitásnál, de nem a képernyõre POKE-al való irásnál/ más készülékre irányitja. Ez egy nyomtató, vagy egy szalagon vagy lemezen levõ adatállomány lehet. Ezt a készüléket vagy állományt elõször meg kell nyitni. A CMD parancsot követnie kell egy számnak vagy numerikus változónak, amelyik az állományra utal.

OPEN 1,4	megnyitja a 4-es készüléket, amelyik a nyomtató
CMD 1	minden normál kimenet most a nyomtatóra megy
LIST	a program listát a nyomtatóra irja és nem a képernyõre.

A képernyõre visszaküldeni a kimenetet a CLOSE 1 állomány lezárással lehet. Elõször egy PRINT# 1-t kell elvégezni az állomány lezárás elõtt.

DATA /adat/

Ezt az utasitást a READ utasitás által használandó tétel lista követi. A tételek lehetnek numerikus értékek vagy szöveges változók és a tételeket vesszõvel kell elválasztani. A szöveges tételeket nem szükséges idézõjelbe tenni, hacsak nem tartalmaznak betüközt, kettõspontot vagy vesszõt. Ha két vessző között nincs semmi akkor Ø-t olvas be számként, vagy egy üres karaktert füzérként.

```
DATA 12, 14.5, "HALLO FIUK", 3.14, RESZ1
```

DEF FN /DEFINE FUNCTION, függvény definiálás/

Ez az utasitás lehetõvé teszi egy összetett számitás

rövid névvel ellátott függvényként való kezelését. Egy hosszu formula esetében, amelyet több helyen használunk egy programon belül, időt és helyet takaritunk meg használatával.

A függvény neve FN és egy tetszőleges megengedett változó név /1 vagy 2 karakter hosszu/. Először a függvényt kell definiálni a DEF utasitással, amelyet követ a függvény név. Egy zárójelbe zárt numerikus változó követi a nevet. Ezután következik a tényleges formula, amit Ön definiálni akar, a megfelelő helyen levő változóval. Ezután lehet "hivni" a formulát, a változó helyére tetszőleges számot téve.

```
1Ø DEF FNA(X) = 12*(34.75 - X/.3)
2Ø PRINT FNA(7)
```

7 kerül a formulában levö X helyére

E példa eredménye 137 lesz.

DIM /DIMENSION, dimenzió/

Amikor 11-nél több elemü tömböt használ, akkor a tömbre egy DIM utasitást kell végrehajtani. Vegye figyelembe, hogy a teljes tömb foglal helyet a tárban, igy ne létesitsen a szükségesnél nagyobb tömböt. A DIM-mel létrehozott változók számát ugy kapjuk meg, hogy összeszorozzuk a tömbök egyes dimenziójának elemszámát.

```
1Ø DIM  A$(40), B7(15), CC%(4,4,4)
```

41 elem 16 elem 125 elem

A DIM utasitásban egynél több tömbnek lehet dimenziót adni. Azonban vigyázzon arra, hogy ne adjon egynél többször dimenziót egy tömbnek.

Ha először CLR van végrehajtva, akkor lehet egy tömböt ujra dimenzionálni. Mindamellett ez kitörli az összes már létrehozott változót.

END /vége/

Amikor a program az END utasitásra kerül, akkor a prog-

ram leáll, mintha kifutott volna a sorokból. A programot CONT-tal ujra lehet inditani.

FOR...TO...STEP /...-tól...-ig....lépés/

Ez az utasitás a NEXT utasitással együtt müködik, megismétel egy programrészt, egy beállitott számu szor. Formája: FOR /ciklusváltozó/ = /számlálás kezdő értéke/ TO /számlálás végértéke/ STEP /lépésköz/. A ciklus változó elősżör felveszi a kezdő értéket, majd a lépésköz értékével növekszik. Lépésköz, vagyis STEP megadás nélkül, a lépésköz 1 lesz. A kezdő és végérték meghatározza a ciklusváltozó határértékeit.

```
10 FOR L = 1 TO 10 STEP .1
20 PRINT L
30 NEXT L
```

A végértéket a STEP szó és egy további szám vagy változó követheti. Ebben az esetben 1 helyett mindig a STEP-et követő értéket adjuk hozzá.Ez lehetővé teszi a visszafelé vagy törtrészenkénti számlálást.

GET /elkap/

A GET utasitás lehetővé teszi a billentyüzetről való adatbeolvasást, egyidejüleg 1 karaktert.Amikor a GET végrehajtásra kerül, a begépelt karakter hozzárendelődik egy változóhoz. Ha nincs karakter begépelve, akkor egy null /üres/ karakter rendelődik hozzá.

A GET-t egy változó név követi, rendszerint egy füzérváltozó.Ha egy numerikus változót használt és egy nem numerikus billentyü lett leütve, akkor a program hibaüzenettel leáll. A GET utasitás elhelyezkedhet egy ciklusban, az üres eredmény ellenőrzésére. Ez a ciklus egy billentyü beütéséig folytatódik.

```
10 GET A$: IF A$ ="" THEN 10
```

GET#

A GET# utasitás egy előzetesen megnyitott készülékkel vagy adatállománnyal kerül felhasználásra azért,

hogy a készülékről vagy állományról egyszerre egy karaktert bevigyen.

GET#1,A$

Ez egy karaktert fog beolvasni az adatállományból.

GOSUB /GOTO SUBROUTINE, menj a szubrutinhoz/

Ez az utasitás azonos a GOTO-val, kivéve abban, hogy ez esetben a számitógép emlékezik arra, hogy a GOSUB utasitás melyik programsorban volt. Amikor egy olyan sor kerül végrehajtásra, mely a RETURN utasitást tartalmazza, a program visszaugrik a GOSUB-ot közvetlenül követő utasitásra. Ez akkor hasznos, amikor van az Ön programjában egy programrészlet, vagy más néven rutin, amelyik a program különböző részeiben fordul elő. A rutin soronkénti begépelése helyett hajtson végre egy-egy GOSUB-ot ahányszor szükség van a rutinra.

```
20 GOSUB 800
```

GOTO vagy GO TO /menj/

Amikor egy GOTO utasitás végrehajtására kerül sor, a következő végrehajtandó sor az lesz, melynek sorszáma azonos a GOTO szót követő számmal.

IF...THEN /ha... akkor/

IF...THEN esetén a számitógép elemez egy helyzetet, és az eredménytől függően, két lehetséges tevékenység közül választ. Ha a kifejezés igaz, akkor a THEN-t követő utasitást hajtja végre. Ez lehet bármilyen BASIC utasitás.
Ha a kifejezés hamis, a program közvetlenül a következő sorra megy.

A kiértékelendő kifejezés változó vagy képlet is lehet, ez esetben igaznak tekintendő, ha nem Ø és hamisnak, ha Ø az értéke. Legtöbb esetben, a kifejezés relációs müveleteket tartalmaz / =, < , > , <=, >=, <>, AND, OR, NOT/.

```
10 IF X > 10 THEN END
```

•

INPUT /bemenet/

Az INPUT utasitás lehetővé teszi, hogy a program a felhasználótól kapjon adatot, hozzárendelve az adatot egy változóhoz. A program leáll, egy kérdőjelet ir a képernyőre és vár a felhasználóra, hogy gépelje be a választ, és üsse be a *RETURN*-t.

Az INPUT-ot egy változónév vagy vesszőkkel elválasztott változó nevek listája követi. Egy üzenet "elhelyezhető" a beviendő változónevek listája előtt azoktól pontosvesszővel elválasztva. Ha egynél több változó kerül bevitelre, akkor begépelésnél vesszőkkel kell elválasztani azokat.

```
10 INPUT "KEREM A CSALAD NEVET"; A$

20 PRINT "A SZEMELYI SZAMAT IRJA BE";: INPUT B
```

INPUT #

Az INPUT# a INPUT-tal megegyezik, csak az előzőleg megnyitott adatállományból vagy készüléktől veszi át az adatot.

```
10 INPUT#1,A
```

LET /legyen/

A LET szót aligha használják programokban, mivel nem kötelező, azonban az utasitás minden BASIC program lényege: A változó név amelyhez a számolás eredményét akarjuk rendelni az egyenlőség jel bal oldalán van, mig a képlet a jobb oldalon.

```
10 LET A = 5

20 LET D$ = "HELLO"
```

NEXT /következő/

A NEXT mindig a FOR utasitással összefüggésben jelenik meg. Amikor a program a NEXT-hez ér, a ciklusváltozót a lépésközzel megnöveli, majd ellenőrzi a FOR utasitást, hogy a végértéket tullépte e már. Ha még nem, akkor a ciklus ujra lefut. Ha igen, akkor a ciklus véget ért, és a NEXT-et követő utasitásnál folytatódik a végrehajtás.

A NEXT-et egy változó név vagy egy vesszőkkel elválasztott változónév lista követheti. Ha nincs megadva változónév akkor az utoljára elinditott ciklus az, amelyre vonatkozik a NEXT. Ha több változó van megadva, akkor balról jobbra érvényesiti azokat.

```
10 FOR X = 1 TO 100: NEXT
```

ON /...től függően/

Ez az utasitás a GOTO és GOSUB utasitásokat az IF utasitás speciális változatává alakitja. Az ON-t egy kifejezés követi, amely kiértékelésre kerül. Ha a számitás eredménye 1, akkor a GOTO vagy GOSUB utáni szám listában az első számnak megfelelő sor lesz végrehajtva, ha 2, akkor a másodiknak megfelelő, és igy tovább. Ha az eredmény 0, negativ vagy nagyobb mint a listában lévő számok száma, akkor az ON-t követő utasitás lesz végrehajtva.

```
10 INPUT X

20 ON X GOTO 10,20,30,40,50
```

OPEN /nyit/

Az OPEN utasitás lehetővé teszi, hogy a Commodore 64 hozzáférjen olyan eszközökhöz, mint a kazettás szalagegység és a lemez egység adatokért, nyomtatóhoz vagy képernyőhöz. Az OPEN után egy szám következik /0-255/, amelyre az összes következő utasitás majd hivatkozik. Rendszerint egy második szám is követi, ez a készülékszám.

A készülék számok a következők:

Ø	képernyő
1	kazetta
4	nyomtató
8	lemez

Lehetséges egy harmadik szám is, ugyancsak vesszővel elválasztva, amely egy másodlagos cim. Kazetta esetében ez olvasásra Ø, irásra 1 és 2 ha szalagvég jelet akarunk frni.

Lemeznél ez a szám a puffer tár vagy a csatorna számát adja. Nyomtatónál a másodlagos cim vezérli a különböző lehetőségeket, mint pl. a széles karakterirást. A részleteket illetően az egyes egységek felhasználói utasitásai adnak felvilágositást.

1Ø OPEN 1,Ø	a képernyőt, mint készüléket megnyitja
2Ø OPEN 2,1,Ø,"D"	megnyitja olvasásra a kazettás egységet és keresi a D állományt
3Ø OPEN 3,4	megnyitja a nyomtatót
4Ø OPEN 4,8,15	megnyitja a lemezen a adatcsatornát.

Lásd még: CLOSE, CMD, GET#, INPUT# es PRINT#

utasitásokat, ST rendszerváltozót és a B függeléket.

POKE /döf, bök/

A POKE után mindig 2 szám vagy kifejezés következik. Az első egy tár hely cime, a második egy Ø és 255 közötti decimalis szám, amely az adott tárhelyre lesz elhelyezve, felülirván bármely korábban tárolt adatot.

```
1Ø POKE 53281,Ø
```

```
2Ø S=4Ø96*13

3Ø POKE S+29,8
```

PRINT /irj[1]/

A PRINT az az utasitás, amelyet a legtöbb ember elõször tanul meg használni, de van számos változata, amiről tudni kell. A PRINT után következhetnek:

- idézõjelbe tett szövegfüzérek
- változónevek
- függvények
- irásjelek

Az irásjelek használata segiti az adatok elhelyezését /formázását/ a képernyõn. A vesszõ a képernyõt négy zónára osztja, mig a pontosvessző elnyom minden szóközt. Mindkét jel a sor végén is állhat. Ez azt eredményezi, hogy a következõ kiirandó dolog ugy lesz kiirva, mintha az elõzõ PRINT utasitásban annak a folytatásaként szerepelnek.

```
1Ø PRINT "HELLO"

2Ø PRINT "HELLO" ,A$

3Ø PRINT A;B

4Ø PRINT J;

6Ø PRINT A,B,C,D
```

Lásd még: POS, SPC és TAB függvényeket.

PRINT#

Néhány különbség van ezen utasitás és a PRINT utasitás között. A PRINT# -t egy szám követi, amelyik az elõzõleg megnyitott készülékre, vagy adatállományra utal.

1/ Szó szerint 'nyomtass'

Ezt a számot egy vessző és a kiirandó lista követi. A vessző és a pontosvessző ugyanolyan hatásu, mint a PRINT-nél. Jegyezzük meg, hogy néhány készülék nem tudja használni a TAB-ot és a SPC-t.

```
100 PRINT#1,"ADATOK ERTEKEI"; A%, B1, C$
```

READ /olvass/

A READ utasitás változókhoz rendeli hozzá a DATA utasitásban lévő információkat, igy az információk felhasználhatóvá válnak. Figyelmet kell forditani arra, hogy ha a READ számot vár, ne füzért olvasson be, amikoris TYPE MISMATCH ERROR /tipus nem egyezés hiba/ üzenet kerül kiirásra.

REM /REMARK, megjegyzés/

A megjegyzés a program listát olvasóknak szól. Magyarázatot füzhet egy programrészlethez, vagy további instrukciókat ad. A REM utasitások semmiféle módon nem befolyásolják a program végrehajtását, csak a program hosszát növelik. A REM után bármilyen szöveg következhet.

RESTORE /visszaáll/

Amikor egy programban hajtjuk végre, akkor az a mutató, amely a DATA utasitásban lévő, a READ által éppen beolvasásra kerülő tételre mutat visszaállitásra kerül a lista első tételére. Ez lehetőséget ad az információk ujrabeolvasására. A RESTORE utasitás önmagában áll egy sorban.

RETURN /menj vissza/

Ez az utasitás mindig a GOSUB-al együtt használatos. Amikor a program egy RETURN utasitáshoz ér, akkor a GOSUB utasitást közvetlenül követő utasitás következik sorra. Ha GOSUB nem volt előzőleg kiadva, akkor RETURN WITHOUT GOSUB ERROR /RETURN GOSUB nélkül hiba/ hibajelzés kerül kiirásra.

STOP /állj/

Ez az utasitás megállitja a program végrehajtását. BREAK IN xxx /xxx sorban megszakitva/ üzenetet ir ki, ahol xxx a STOP-ot tartalmazó sor száma. A programot ujra lehet inditani a CONT utasitással. A STOP-ot rendszerint a program belövésénél használjuk.

SYS /SYSTEM, rendszer/

A SYS után egy decimális szám vagy numerikus érték következik a 0-65535 tartományon belül. A program az adott tárhelytől kezdődően végrehajtja az ott kezdődő gépi nyelvü programot. Ez hasonlit a USR függvényre, de nem enged paraméter átadást.

WAIT /várj/

A WAIT arra szolgál, hogy megállitsa a programot mindaddig, mig egy tárhely tartalma ki nem cserélődik megadott módon. A WAIT-et egy tárhely /X/ és egy vagy két változó követi. Formája:

WAIT X,Y,Z

Először a tárhely tartalma és a harmadik változó között /ha van/ kizáró VAGY logikai müvelet kerül végrehajtásra, majd az eredmény és a második változó között ÉS logikai müvelet jön létre. Ha az eredmény Ø, akkor a program visszamegy a tárhelyre és ujra ellenőrzi. Amikor az eredmény nem nulla, a program a következő utasitáson folytatódik.

NUMERIKUS FÜGGVÉNYEK

ABS(X) /abszolut érték/

Az ABS a szám abszolutértékét adja vissza, előjel nélkül /+ vagy - /. Az eredmény mindig pozitiv.

ATN(X) /arkusz tangens/

Azt a radiánban megadott szöget adja vissza, amelynek

tangense X.

COS(X) /Koszinusz/

X koszinuszának értékét adja vissza, ahol az X szög radiánban van megadva.

EXP(X) /exponenciális függvény/

Visszaadja az e/2.71827183/ matematikai állandó X-ik hatványának értékét.

FNxx(X)

A DEFFNxx(X) utasitással a felhasználó által meghatározott xx függvény értékét adja vissza.

INT(X) /egész rész/

Az X értékének levágott értékét, azaz decimális ponttól jobbra eső rész nélkül adja vissza. Az eredmény mindig kisebb vagy egyenlő X-el. Igy a negativ, tizedesszámokat is tartalmazó számból az eredeti értéknél kisebb szám lesz.

LOG(X) /logaritmus/

X természetes alapu logaritmusát adja vissza. A természetes alap: e /lásd EXP(X)). lO alapu logaritmusra váltani LOG (lO)-el való osztással lehet.

PEEK(X) /les/

Az X tárhely tartalmát adja meg O-65535 értéktartományon belül és O-255 közötti eredményt ad. A PEEK gyakran használatos a POKE utasitással együtt.

RND(X) /véletlen szám/

RND(X) véletlen számot állit elő a O-l tartományban. Az első véletlen számot az RND(-TI) kifejezéssel kell létrehozni ahhoz, hogy mindig különböző módon induljon. Ezután X-nek l-nek vagy egy pozitiv számnak kell lennie. Ha X Ø, akkor az eredmény ugyanaz a véletlen szám

lesz, mint az előző volt.

Negativ X érték ujrainditja a véletszámgenerátort. Ugyanannak a negativ számnak a használata ugyanazt a véletlen szám sorozatot eredményezi.

X és Y közötti szám létrehozására a képlet:

```
N= INT((Y-X+1)*RND (1))+X
```

ahol

Y a felső határ

X a kivánt számok értékkészletének alsó határa.

SGN(X) /előjel/

A függvény X előjelét /pozitiv, negativ, vagy nulla/ adja vissza. Az eredmény +1, ha pozitiv, Ø ha nulla és -1, ha negativ.

SIN(X) /szinusz/

SIN(X) a trigonometriai szinusz függvény. Az eredmény X szinusza, ahol X egy szög radián van.

SQR(X) /négyzetgyök/

Ez a függvény X négyzetgyökét adja meg, ahol X pozitiv szám, vagy Ø. Ha X negativ, akkor az eredmény ILLEGAL QUANTITY ERROR /nem megengedett mennyiség hiba/ hiba-jelzés.

TAN(X) /tangens/

X tangensét adja eredményül, ahol X radiánban megadott szög.

USR(X) /felhasználói szubrutin/

Amikor ezt a függvényt használjuk a program egy gépi nyelvü programra ugrik, amelynek elindulási cime a tár rögzitett helyén található. Az X paraméter átadásra

kerül a gépi nyelvű program részére, amely a BASIC programba való visszatéréskor egy másik értéket fog átadni. Erre a függvényre és a gépi nyelvű programozásra vonatkozó részletesebb ismertetést a Commodore 64 Programozói Kézikönyvben talál.

FÜZÉR FÜGGVÉNYEK

ASC(X$) vagy ASC("A") /ASCII függvény/

Ez a függvény X$ első karakterének ASCII kódját adja vissza.

CHR$(X) /karakter függvény/

Ez az ASC-nek ellentéte, egy karaktert ad vissza, amelynek ASCII kódja X.

LEFT$(X$,X) /balrész függvény/

Vissza ad egy füzést, amely X$-nek a balról vett X db karakterét tartalmazza.

LEN(X$) /hossz függvény/

Az X$ karaktereinek /beleértve a szóközöket és egyéb szimbólumokat/ számát adja meg.

MID$(X$,S,X) /középrész függvény/

Vissza ad egy füzért, amely X$-nek az S-edik karakterétől számitott X db karakterét tartalmazza.

Ha X nincs megadva, akkor az összes karaktert tartalmazza az S-ediktől végig.

RIGHT$(X$,X) /jobbrész függvény/

Vissza ad egy füzért, amely X$-nek a jobbról számitott X db karakterét tartalmazza.

STR$(X) /füzér függvény/

Egy füzért ad vissza, amely az X numerikus érték karaktert füzér változata.

VAL(X$) /érték függvény/

Ez a függvény X$-t átalakitja egy számmá, és alapvetõen az STR$ forditott müvelete. A füzér legbalolda-libb karaktertõl kerül vizsgálatra, és annyi karakter lesz figyelembe véve ahány számként felismerhetõ.

```
10 X=VAL("123.456")          X=123.456
10 X=VAL("12A13B")           X=12
10 X=VAL("RIU017")           X=0
10 X=VAL("-1.23.45.67")      X=-1.23
A$="1234":X=VAL(A$):X=1234
```

EGYÉB FÜGGVÉNYEK

FRE(X) /szabad hely/

Ez a függvény megadja a tárban rendelkezésre álló felhasználatlan byte-ok számát, X értékétõl függetlenül.

POS(X) /pozició/

Ez a függvény annak az oszlopnak a számát /0-39/ adja vissza, amelynél a következõ PRINT utasitás irni fog kezdeni a képernyõn. X bármilyen érték lehet, és nincs használva.

SPC(X) szóköz/

Ezt a PRINT utasitásban használva X betüközt hagy ki.

TAB(X) /tabulátor/

Szintén a PRINT utasitásban használatos, az X oszlopban lesz a következõ tétel kiirva,hacsak az aktuális kiirási helyzet nem lépte még tul ama helyet.

BASIC KULCSSZAVAK RÖVIDITÉSEI

Programok és parancsok begépelésénél idősporolás végett a COMMODORE 64 BASIC lehetővé teszi a legtöbb kulcsszó rövidítését. A PRINT rövitésére szolgál a kérdőjel. Egyéb szavak rövidítését a szó első egy vagy két betüje, majd a *SHIFT* és a következő betü adja. Ha programsorban használunk rövidítéseket, a LIST-tel kapott listában teljes szó jelenik meg. Megjegyezzük, hogy néhány rövidített kulcsszó bal zárójelet is tartalmaz.

Parancs	Rövidítés	Megjelenés	Parancs	Rövidítés	Megjelenés
ABS	A SHIFT B	A	FOR	F SHIFT O	F
AND	A SHIFT N	A	FRE	F SHIFT R	F
ASC	A SHIFT S	A	GET	G SHIFT E	G
ATN	A SHIFT T	A	GOSUB	GO SHIFT S	GO
CHR$	C SHIFT H	C	GOTO	G SHIFT O	G
CLOSE	CL SHIFT O	CL	INPUT#	I SHIFT N	I
CLR	C SHIFT L	C	LET	L SHIFT E	L
CMD	C SHIFT M	C	LEFT$	LE SHIFT F	LE
CONT	C SHIFT O	C	LIST	L SHIFT I	L
DATA	D SHIFT A	D	LOAD	L SHIFT O	L
DEF	D SHIFT E	D	MID$	M SHIFT I	M
DIM	D SHIFT I	D	NEXT	N SHIFT E	N
END	E SHIFT N	E	NOT	N SHIFT O	N
EXP	E SHIFT X	E	OPEN	O SHIFT P	O

Parancs	Rövidités	Megjelenés
PEEK	P SHIFT E	P
POKE	P SHIFT O	P
PRINT	?	?
PRINT#	P SHIFT R	P
READ	R SHIFT E	R
RESTORE	RE SHIFT S	RE
RETURN	RE SHIFT T	RE
RIGHT$	R SHIFT I	R
RND	R SHIFT N	R
RUN	R SHIFT U	R
SAVE	S SHIFT A	S
SGN	S SHIFT G	S
SIN	S SHIFT I	S

Parancs	Rövidités	Megjelenés
SPC(	S SHIFT P	S
SQR	S SHIFT Q	S
STEP	ST SHIFT E	ST
STOP	S SHIFT T	S
STR$	ST SHIFT R	ST
SYS	S SHIFT Y	S
TAB	T SHIFT A	T
THEN	T SHIFT H	T
USR	U SHIFT S	U
VAL	V SHIFT A	V
VERIFY	V SHIFT E	V
WAIT	W SHIFT A	W

KÉPERNYŐ KÓDOK

A következő táblázat a Commodore 64 karakter készletbe rögzitett összes karaktert tartalmazza. Azt a számot mutatja, amelyet a képernyő tárba /1024-2023 rekeszek/ kell POKE -kal bevinni egy kivánt karakter megjelenitéséhez. Azt is mutatja, hogy milyen karakter felel meg a képernyőről PEEK-kel kiolvasott számnak.

Két karakterkészlet áll rendelkezésre, de egyidőben csak egyet használunk. Ez azt jelenti, hogy nem lehet az egyik ill. a másik készletből való karaktereket egyidejüleg a képernyőn megjeleniteni. A *SHIFT* és *COMMODORE* nyomógombok egyidejü lenyomásával lehet a készleteket kapcsolni.

BASIC-ből a POKE 53272,29 a nagybetü/grafikus módot és a POKE 53272,31 a kisbetü/nagybetü módot kapcsolja.

A táblázat bármely száma inverz módban is kiirható. Az inverz kódot ugy kaphatjuk meg, ha a táblázatbeli értékhez 128-t hozzáadunk.

Ha egy tömör kört akar az 1504-s helyen kiiratni, akkor POKE 1504,81-t kell irni, mivel a kör kódja 81.

Van egy megfelelő tárhely, amely vezérli a képernyőn az egyes karakterek színét /55296-56295 rekeszek/. A kör színét sárgára /a szin kódja 7/ váltani POKE-kal lehet ugy, hogy a megfelelő tárhelyre /55776/ beirjuk a karakter színt: POKE 55776,7.

A G függelékben található a teljes képernyő és szintár-térkép leirás a szín kodokkal együtt.

1.MÓD	2.MÓD	KÓD	1.MÓD	2.MÓD	KÓD	1.MÓD	2.MÓD	KÓD
@	∅	0	V	v	22	,		44
A	a	1	W	w	23	-		45
B	b	2	X	x	24	.		46
C	c	3	Y	y	25	/		47
D	d	4	Z	z	26	0		48
E	e	5	[		27	1		49
F	f	6	£		28	2		50
G	g	7	]		29	3		51
H	h	8	↑		30	4		52
I	i	9	←		31	5		53
J	j	10	SPACE		32	6		54
K	k	11	!		33	7		55
L	l	12	"		34	8		56
M	m	13	#		35	9		57
N	n	14	$		36	:		58
O	o	15	%		37	;		59
P	p	16	&		38	<		60
Q	q	17	'		39	=		61
R	r	18	(		40	>		62
S	s	19	)		41	?		63
T	t	20	*		42			
U	u	21	+		43			

Megjegyzés: 32 a SZÓKÖZ kódja

1.MÓD	2.MÓD	KÓD
		64
	A	65
	B	66
	C	67
	D	68
	E	69
	F	70
	G	71
	H	72
	I	73
	J	74
	K	75
	L	76
	M	77
	N	78
	O	79
	P	80
	Q	81
	R	82
	S	83
	T	84
	U	85

1.MÓD	2.MÓD	KÓD
	V	86
	W	87
	X	88
	Y	89
	Z	90
		91
		92
		93
		94
		95
SPACE		96
		97
		98
		99
		100
		101
		102
		103
		104
		105
		106
		107

1.MÓD	2.MÓD	KÓD
		108
		109
		110
		111
		112
		113
		114
		115
		116
		117
		118
		119
		120
		121
		122
		123
		124
		125
		126
		127

Megjegyzés: 96 a SHIFT+SZÓKÖZ kódja

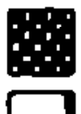
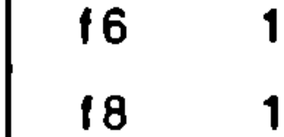
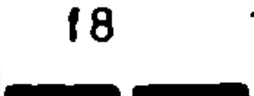
A 128-255 kódok a 0-127 kódok inverz képei

ASCII ÉS CHR$ KÓDOK

Ez a függelék megmutatja, hogy X lehetséges értékeire a PRINT CHR$(X) milyen karaktereket jelenit meg. Ugyancsak mutatja azt, hogy PRINT ASC("x") begépelésekor, ahol x bármely, a billentyüzetről begépelhető karakter, milyen értéket kapunk. Mindez hasznos a GET utasitással kapott karakterek kiértékelésénél, a kis és nagybetük közti konvertálásnál és olyan karaktereken alapuló parancsok PRINT-ben való megadásakor, amelyeket nem lehet idézőjelbe tenni.

KÉP	CHR$	KÉP	CHR$	KÉP	CHR$	KÉP	CHR$
	0	CRSR	17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	HOME	19	$	36	5	53
	3	DEL	20	%	37	6	54
	4		21	&	38	7	55
	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
DISABLES	8		25	*	42	;	59
ENABLES	9		26	+	43	<	60
	10		27	,	44	=	61
	11	RED	28	-	45	>	62
	12	CRSR	29	.	46	?	63
RETURN	13	GRN	30	/	47	@	64
	14	BLU	31	0	48	A	65
	15	SPACE	32	1	49	B	66
	16	!	33	2	50	C	67

Megjegyzések: 8 blokkolja a mód váltást
9 engedélyezi a mód váltást
14 átkapcsol kisbetűre
32 a szóköz kódja

KÉP	CHR$	KÉP	CHR$	KÉP	CHR$	KÉP	CHR$
D	68		97		126		155
E	69		98		127		156
F	70		99		128		157
G	71		100		129		158
H	72		101		130		159
I	73		102		131		160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163
M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	f8	140		169
S	83		112		141		170
T	84		113		142		171
U	85		114		143		172
V	86		115		144		173
W	87		116		145		174
X	88		117		146		175
Y	89		118		147		176
Z	90		119		148		177
[	91		120		149		178
£	92		121		150		179
]	93		122		151		180
↑	94		123		152		181
←	95		124		153		182
	96		125		154		183

Megjegyzések: 142 átkapcsol nagybetüre
160 a shift+szóköz kódja

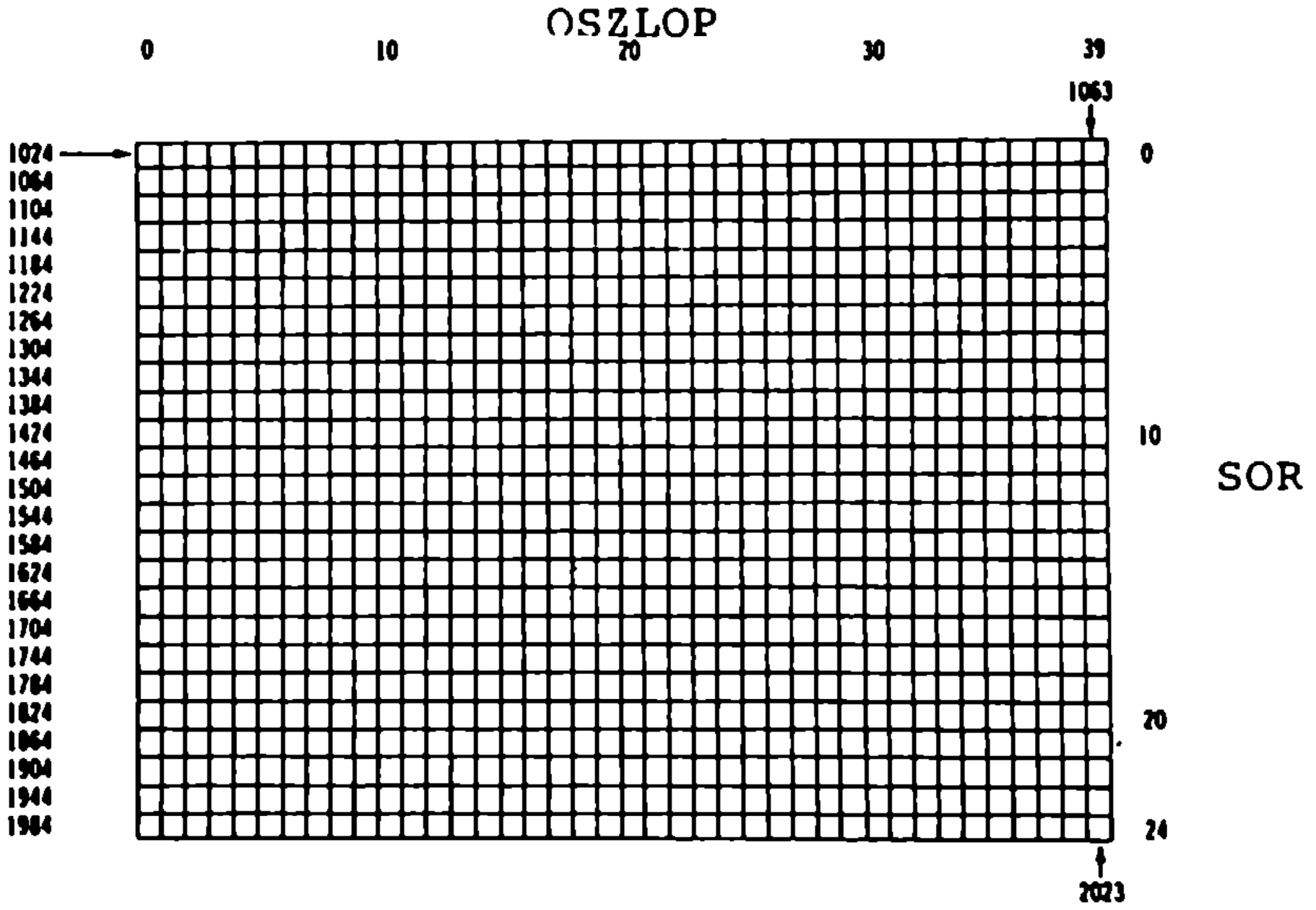
KÉP	CHR$	KÉP	CHR$	KÉP	CHR$	KÉP	CHR$
	184		186		188		190
	185		187		189		191

Megjegyzések: 192-223 kódok azonosak a 96-127 kódokkal
224-254 " " 160-190 "
255 kód azonos a 126 kóddal

A KÉPERNYŐ ÉS A SZÍNEK TÁRTÉRKÉPEI

A következő táblázatokon látható, hogy mely tárhelyek vezérlik a képernyő adott helyein elhelyezhető karaktereket, illetve melyek vezérlik ugyanezen karakterek szinét. Ugyancsak megadjuk a karakterek szinének kódjait.

KÉPERNYŐ TÁRTÉRKÉP

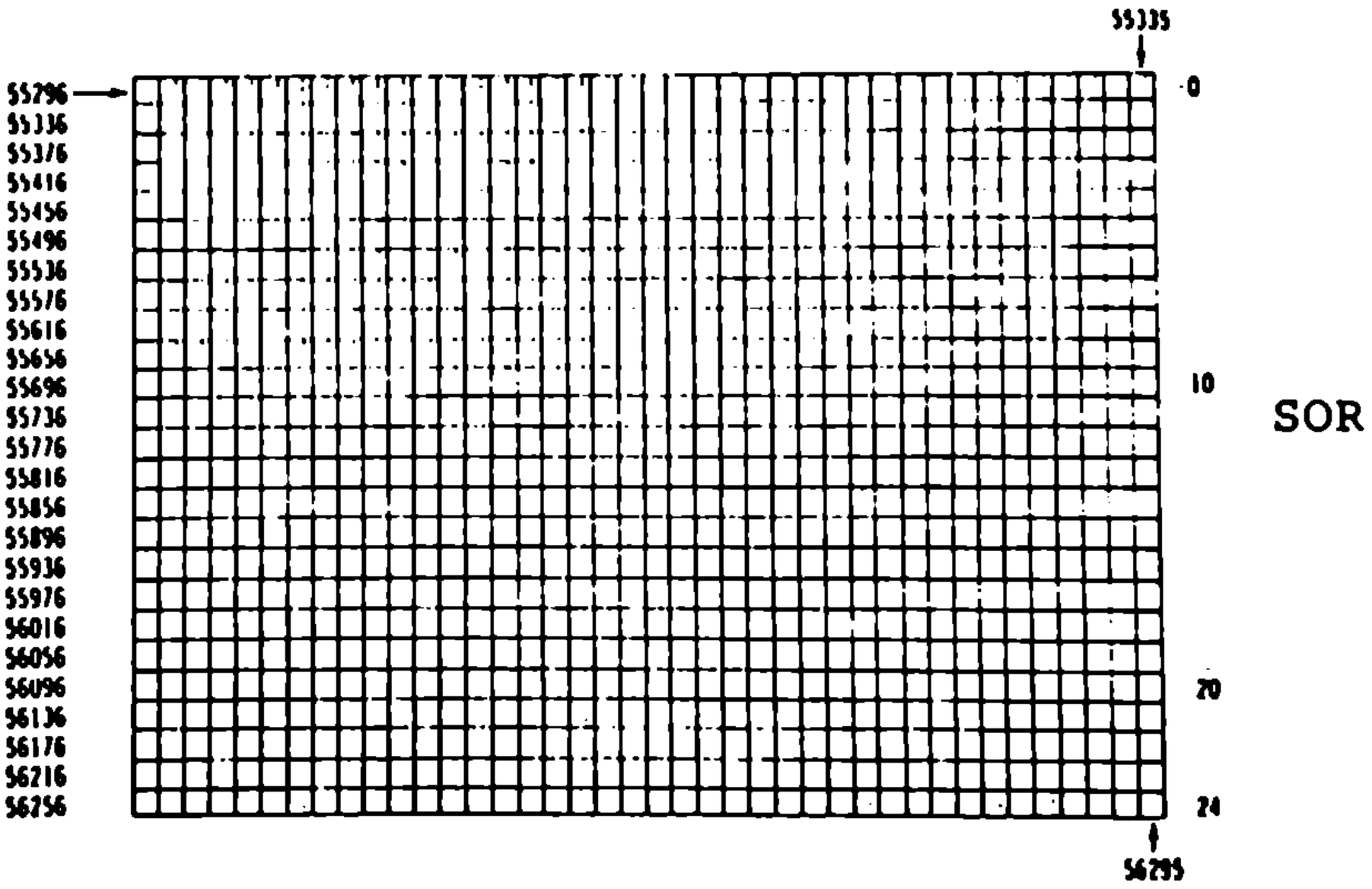


A karakterek színét a következőkre lehet változtatni, ha a megadott értéket POKE utasitással elhelyezzük a szín tár megfelelő helyére:

Ø	fekete	8	narancs
1	fehér	9	barna
2	piros	10	világospiros
3	türkiz	11	sötétszürke
4	lila	12	középszürke
5	zöld	13	világoszöld
6	kék	14	világoskék
7	sárga	15	világosszürke

Például ha a képernyõ bal felsõ sarkában lévõ karakter szinét pirosra akarjuk cserélni, akkor a következõt irjuk be: POKE 55296,2.

SZIN TÁR TÉRKÉP



SZÁMITOTT MATEMATIKAI FÜGGVÉNYEK

Azok a függvények, amelyek a COMMODORE 64 BASIC-nek nem részei a következő módon számithatók ki:

FÜGGVÉNYEK	BASIC EKVIVALENS
SEKANS	SEC(X) = 1/COS(X)
COSEKANS	CSC(X) = 1/SIN(X)
COTANGENS	COT(X) = 1/TAN(X)
ARCUSSINUS	ARSIN(X) = ATN(X/SQR(1-X↑2))
ARCUCOSINUS	ARCOS(X) = -ATN(X/SQR(1-X↑2))+π/2
ARCUSCOTANGENS	ARCOT(X) = ATN(X)+π/2
ARCUSSEKANS	ARSEC(X) = ATN(X/SQR(X↑2-1))
ARCUSCOSECANS	ARCSC(X) = ATN(X/SQR(X↑2-1)) +(SGN(X)-1)·π/2
SINUS HYPERBOLICUS	SINH(X) = (EXP(X)-EXP(-X))/2
COSINUS HYPERBOLICUS	COSH(X) = (EXP(X)+EXP(-X))/2
TANGENS HYPERBOLICUS	TANH(X) = EXP(-X)/(EXP(X)+EXP (-X))·2+1
CONTANGENS HYPERBOLICUS	COTH(X) = EXP(-X)/(EXP(X)-EXP (-X))·2+1
SEKANS HYPERBOLICUS	SECH(X) = 2/(EXP(X)+EXP(-X))
COSEKANS HYPERBOLICUS	CSCH(X) = 2/(EXP(X)-EXP(-X))
ARCUSSINUS HYPERBOLICUS	ARSINH(X) = LOG(X+SQR(X↑2+1))
ARCUSCOSINUS HYPERBOLICUS	ARCOSH(X) = LOG(X+SQR(X↑2-1))
ARCUSTANGENS HYPERBOLICUS	ARTANH(X) = LOG((1+X)/(1-X))/2
ARCUSCOTANGENS HYPERBOLICUS	ARCOTH(X) = LOG((X+1)/(X-1))/2
ARCUSSEKANS HYPERBOLICUS	ARSECH(X) = LOG((SQR(1-X↑2)+1)/X)
ARCUSCOSEKANS HYPERBOLICUS	ARCSCH(X) = LOG((SQR(1+X↑2)+1)/X) ·SGN(X)

I FÜGGELÉK

CSATLAKOZÓ KIOSZTÁS BE- ÉS KIMENETI ESZKÖZÖK RÉSZÉRE

Ez a függelék mutatja meg, hogy milyen csatlakozásokat lehet a COMMODORE 64-hez készíteni.

1/ Játék be- és kimenet

2/ Modul csatlakozó

3/ Hang/kép

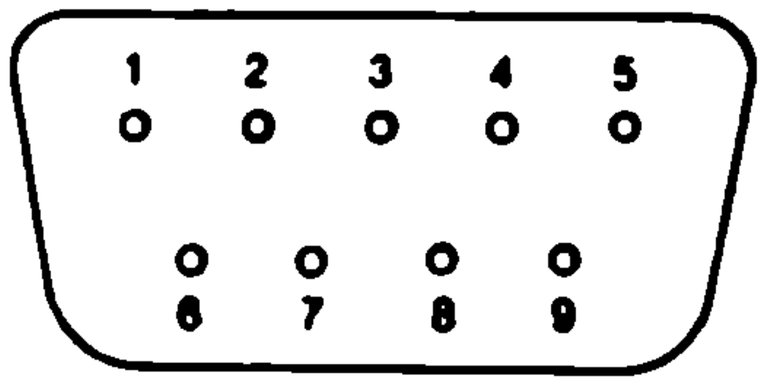
4/ Soros be/kimenet /Lemez/nyomtató/

5/ Kazetta

6/ Felhasználói interfész

Játék be/ki 1

PÓLUS	JEL	MEGJ.
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY**	
6	BUTTON A/LP*	
7	+5V	MAX. 100mA
8	GND	
9	POT AX**	



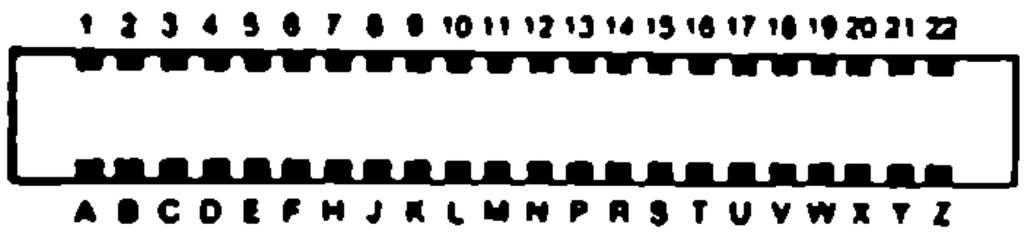
Játék be/ki 2

PÓLUS	JEL	MEGJ.
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY**	
6	BUTTON B	
7	+5V	MAX. 100mA
8	GND	
9	POT BX**	

Modul csatlakozó

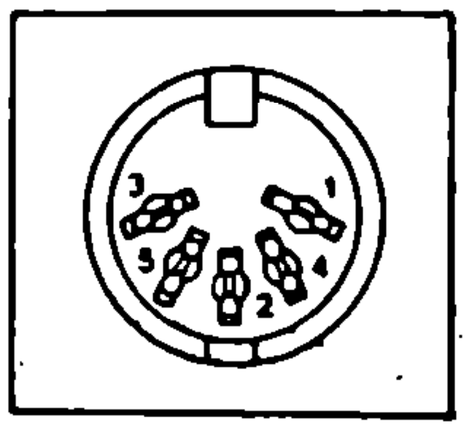
PÓLUS	JEL
22	GND
21	CD0
20	CD1
19	CD2
18	CD3
17	CD4
16	CD5
15	CD6
14	CD7
13	$\overline{\text{DMA}}$
12	BA
11	$\overline{\text{ROML}}$
10	I/O2
9	$\overline{\text{EXROM}}$
8	$\overline{\text{GAME}}$
7	I/O1
6	Dot Clock
5	$\overline{\text{R/W}}$
4	$\overline{\text{IRQ}}$
3	+5V
2	+5V
1	GND

PÓLUS	JEL
Z	GND
Y	CA0
X	CA1
W	CA2
V	CA3
U	CA4
T	CA5
S	CA6
R	CA7
P	CA8
N	CA9
M	CA10
L	CA11
K	CA12
J	CA13
H	CA14
F	CA15
E	$\overline{\text{S02}}$
D	$\overline{\text{NMI}}$
C	$\overline{\text{RESET}}$
B	$\overline{\text{ROMH}}$
A	GND



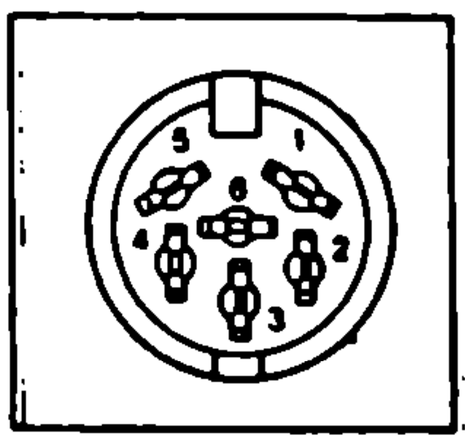
Hang/kép

PÓLUS	JEL	
1	LUMINANCE	
2	GND	
3	AUDIO OUT	
4	VIDEO OUT	
5	AUDIO IN	



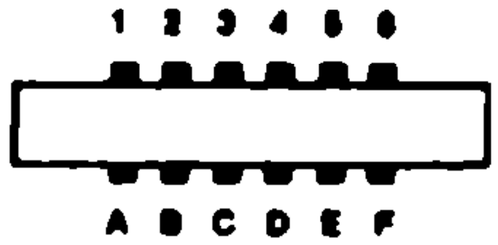
Soros be/ki

PÓLUS	JEL
1	SERIAL $\overline{\text{SRQIN}}$
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	$\overline{\text{RESET}}$



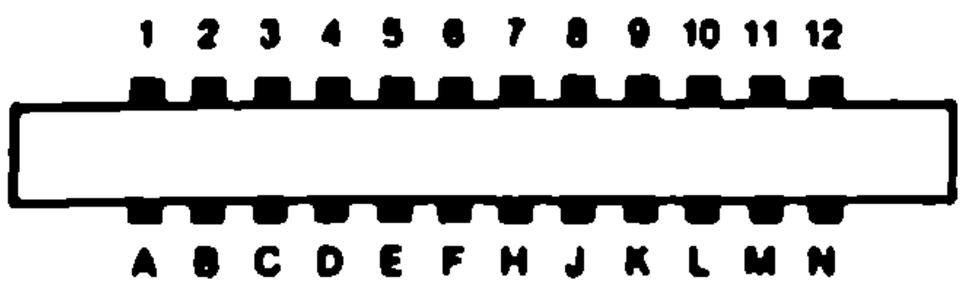
Kazetta

PÓLUS	JEL
A-1	GND
B-2	+5V
C-3	CASSETTE MOTOR
D-4	CASSETTE READ
E-5	CASSETTE WRITE
F-6	CASSETTE SENSE



Felhasználói interfész

PÓLUS	JEL	MEGJEGYZÉS
1	GND	
2	+5V	MAX 100 mA
3	$\overline{\text{RESET}}$	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	$\overline{\text{PC2}}$	
9	SER. ATN IN	
10	9 VAC	MAX. 100 mA
11	9 VAC	MAX. 100 mA
12	GND	
A	GND	
B	$\overline{\text{FLAG2}}$	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	



KIPRÓBÁLANDÓ PROGRAMOK

A következő hasznos programokat próbálja ki COMMODORE 64 gépén. Ezek a programok egyaránt szórakoztatóak és hasznosak.

```
100 PRINT"■■5-BETUS JATEK   JIM BUTTERFIELD"
120 INPUT"■■ZABALYOKAT KERI";Z$:IFASC(Z$)=78GOTO250
130 PRINT"■■PROBALJA ELTALALNI AZ 5-BETUS SZOT"
140 PRINT"■■CSAK EKEZET NELKULI, ERTELMES 5-BETUS"
150 PRINT"FONEVEKET TALALGATHAT."
153 PRINT"HOSSZU I, O ES U MEG VAN ENGEDVE"
160 PRINT"A TALALATOK SZAMAT MEGMONDOM"
180 PRINT"■■TIPP: A TRUKK KEVESET VALTOZTATNI"
190 PRINT"  AZ EGYIK PROBALKOZASTOL A KOVETKEZOIG"
200 PRINT"  PL. HA 'FODOR' 2 TALALAT,PROBALKOZZON"
210 PRINT"  'BODOR'-RAL VAGY 'ROBOT'-TAL"
250 DATA NBSIB,DTVDT,WFSFN,LBQDB,LPDTJ
260 DATA NBMBD,CBSPN,UVEPT,LBMBQ,EVEPS
270 DATA CBOZB,BSBTA,MBCEB,VEWBS,LVUZB
280 DATA DTJLP,DTBMP,FNCFS,CVDVT,PSWPT
290 DATA QPTUB,QBDJT,LBSPN,GPUFM,LPDLB
300 DATA MJTAU,MJUFS,MPWBH,NJOUB,NPLVT
310 DATA GPOBM,HBUZB,HFOOZ,HBSBU,HZVGB
320 DATA IBMPN,NBMPN,LBSPN,IJUFM,IJOUB
330 DATA IPSEP,IPTTA,KPHBS,LBMPA,LBOOB
340 DATA LBNBU,LBNSB,LBSBK,LBSOB,LBQPS
400 N=50
410 DIM N$(N),Z(5),Y(5)
420 FOR J=1TON:READN$(J):NEXTJ
430 T=TI
440 T=T/1000:IFT>=1THENGOTO440
450 Z=RND(-T)
500 G=0:N$=N$(RND(1)*N+1)
510 PRINT "■■VAN EGY 5-BETUS SZAVAM:":IFR>0GOTO560
520 PRINT "TALALGASSON (ERTELMES SZAVAKKAL)"
530 PRINT "ES EN MEGMONDOM HANY BETUT TALALT EL."
540 PRINT "A SORREND NEM SZAMIT"
```

```
560 G=G+1:INPUT "AZ ON SZAVA";Z$
570 IF LEN(Z$)<>5THENPRINT"5-BETUS SZOT KELL BEIRNIA!":
    GOTO560
580 V=0:H=0:M=0
590 FORJ=1TO5
600 Z=ASC(MID$(Z$,J,1)):Y=ASC(MID$(N$,J,1))-1
605 IFY=64THENY=90
610 IFZ<65ORZ>90THENPRINT"EZ NEM EGY SZO!":GOTO560
620 IFZ=65ORZ=69ORZ=73ORZ=79ORZ=85ORZ=89THENV=V+1
630 IFZ=YTHENM=M+1
640 Z(J)=Z:Y(J)=Y:NEXTJ
650 IFM=5GOTO800
660 IFV=0ORV=5THENPRINT"EJNYE...MICSODA SZO EZ?":GOTO560
670 FORJ=1TO5:Y=Y(J)
680 FORK=1TO5:IFY=Z(K)THENH=H+1:Z(K)=0:GOTO700
690 NEXTK
700 NEXTJ
710 PRINT"[illegible]";H;"TALALAT"
720 IFG<30GOTO560
730 PRINT "JOBB LESZ HA MEGMONDOM! A SZO"
735 PRINT "'";
740 FORJ=1TO5:PRINTCHR$(Y(J));:NEXTJ
750 PRINT "'":GOTO810
800 PRINT G;"PROBALKOZASSAL TALALTA EL!"
810 INPUT "XEGY UJABB SZOT";Z$
820 R=1:IFASC(Z$)<>78GOTO500
```

```
1 REM *** SORRENDEZES
2 REM
3 REM *** A PET FELHASZNALOI CSOPORTTOL
4 REM *** SZOFTVER CSERE
5 REM *** PF 371
6 REM *** MONTGOMERYVILLE, PA 18936
7 REM
50 DIM A$(26)
100 Z$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
110 Z1$="12345678901234567890123456"
```

```
200 PRINT"■■■■A RENDEZENDO FUZER HOSSZA?■"
220 INPUT"A MAX HOSSZ 26 ";F%
230 IF F%<1 OR F%>26 THEN 200
240 F=F%
300 FOR I=1 TO F
310 A$(I)=MID$(Z$,I,1)
320 NEXT I
400 REM FUZER OSSZEKEVERESE
420 FOR I=1 TO F
430 K=INT(RND(1)*F+1)
440 T$=A$(I)
450 A$(I)=A$(K)
460 A$(K)=T$
470 NEXT I
480 GOSUB 950
595 T=0
600 REM RESZFUZER MEGFORDITASA
605 T=T+1
610 INPUT"HANYAT IRJAK FORDITOTT SORRENDBEN";R%
620 IF R%=0 GOTO 900
630 IF R%>0 AND R%<=F GOTO650
640 PRINT"1 ES";F;" KOZOTT KELL LENNIE":GOTO 610
650 R=INT(R%/2)
660 FOR I=1 TO R
670 T$=A$(I)
680 A$(I)=A$(R%-I+1)
690 A$(R%-I+1)=T$
700 NEXT I
750 GOSUB 950
800 C=1:FOR I=2 TO F
810 IF A$(I)>A$(I-1) GOTO 830
820 C=0
830 NEXT I
840 IF C=0 GOTO 600
850 PRINT"■";T;"PROBALGATASSAL OLDOTTA MEG"
900 REM UJABB JATEK FELAJANLASA
910 INPUT"■MEG EGY JATEKOT";V$
920 IF LEFT$(V$,1)="I"OR V$="1" GOTO 200
930 END
950 PRINT
960 PRINT LEFT$(Z1$,F)
970 FOR I=1 TO F:PRINT A$(I);:NEXT I
980 PRINT"■"
990 RETURN
```

```
90 REM ZONGORA BILLENTYU
100 PRINT"■ ■ ■ ■ | ■ ■ ■ | ■ ■ | ■ ■ "
110 PRINT" ■ ■ ■ | ■ ■ ■ | ■ ■ | ■ ■ "
120 PRINT" ■ ■ ■ | ■ ■ ■ | ■ ■ | ■ ■ "
130 PRINT" ■ | | | | | | | | | | | "
140 PRINT" ■Q|W|E|R|T|Y|U| I|O|P|@|*|↑"
150 PRINT"■PEDAL(BE/KI): SZOKOZ"
160 PRINT"■OKTAV KIVALASZTAS:'F1,F3,F5,F7'"
170 PRINT"■HULLAMALAK:'F2,F4,F6,F8'■"
180 PRINT"VARJON, MOST KESZULOK FEL"
190 S=13*4096+1024:DIMF(26):DIMK(255)
200 FORI=0TO28:POKES+I,0:NEXT
205 POKES+24,15:REM HANGERO
210 F1=7040:FORI=1TO26:F(27-I)=F1*5.8+30
215 F1=F1/2↑(1/12):NEXT
220 K$="Q2W3ER5T6Y7UI9O0P@-*£↑"
230 FORI=1TOLEN(K$):K(ASC(MID$(K$,I)))=I:NEXT
240 PRINT"■                                    "
250 FE=9:LE=0:TA=15:EL=8:TE=TA*16+EL:FL=FE*16+LE
255 HA=16:W=0:M=1:OK=4:MB=256:Z=0
260 FORI=0TO2:POKES+5+I*7,FL
265 POKES+6+I*7,TE
270 POKES+2+I*7,4000AND255:POKES+3+I*7,4000/256:NEXT
280 GOTO595
300 GETA$:IFA$=""THEN300
310 FR=F(K(ASC(A$)))/M:AR=S+4:IFFR=0THEN300
320 POKES+6,Z:REM TART/ELENGED BEF.
325 POKES+5,Z:REM FELF/LECSENG BEF.
330 POKEAR,8:POKEAR,0:REM FIX BEF.
340 POKES,FR-MB*INT(FR/MB) REM AF BEALL.
350 POKES+1,FR/MB:REM MF BEALL.
360 POKES+6,TE:REM TART/ELENGED BEALL.
365 POKES+5,FL:REM FELF/LECSENG BEALL.
370 POKEAR,HA+1:FORI=1TO50*FL:NEXTI
375 POKEAR,HA:REM IMPULZUS
400 GOTO300
500 IFA$="■"THENM=1:OK=4:GOTO595
510 IFA$="■"THENM=2:OK=3:GOTO595
520 IFA$="■"THENM=4:OK=2:GOTO595
530 IFA$="■"THENM=8:OK=1:GOTO595
540 IFA$="■"THENW=0:HA=16:GOTO595
550 IFA$="■"THENW=1:HA=32:GOTO595
560 IFA$="■"THENW=2:HA=64:GOTO595
570 IFA$="■"THENW=3:HA=128:GOTO595
580 IFA$="  "THENTE=(TEANDNOT2)OR(NOTTEAND2):GOTO300
590 IFA$="■"THEN250
595 PRINT"■OKTAV:";OK;" HULLAMALAK:";W
600 GOTO300
```

HIBAÜZENETEK

Ez a függelék tartalmazza a Commodore 64 által létrehozott hibaüzenetek teljes listáját, a hibát létrehozó magyarázattal együtt.

BAD DATA/Hibás adat/- egy megnyitott adatállományból füzért kapott, de a program numerikus adatot várt.

BAD SUBSCRIPT /Hibás index/- a program egy olyan tömb elemre próbált hivatkozni, amelyik a DIM-ben megadott dimenzión kivül esik.

CAN'T CONTINUE /Nem folytatható/- a CONT parancsot nem tudja végrehajtani azért, mert a program soha nem futott, vagy hibás volt vagy mert egy sor javitásra került.

DEVICE NOT PRESENT /Nem jelenlevő készülék/ - a szükséges bemeneti/kimeneti készülék nem áll az OPEN, CLOSE, CMD, PRINT#, INPUT vagy GET# rendelkezésére.

DIVISION BY ZERO /Nullával való osztás/ - a nullával való osztás nem megengedett.

EXTRA IGNORED /Felesleg visszautasitva/ - tul sok adattétel lett begépelve az INPUT utasitásra adott válaszként. Csak az első néhány tétel került elfogadásra.

FILE NOT FOUND /Nem találta az adatállományt/ - a szalagon egy állományt keresett, és END-OF-FILE /állomány vége/ jelet talált vagy lemez esetén nem létezik a keresett nevü adatállomány.

FILE NOT OPEN /Nincs nyitva az adatállomány/ - a CLOSE, CMD, PRINT#, INPUT# vagy GET# parancsban hivatkozott állomány nem lett előzetesen megnyitva.

FILE OPEN /Adatállomány nyitva/ - kisérlet történt egy már nyitott állomány nyitására.

FORMULA TOO COMPLEX /Tulzottan összetett formula/ - a kiértékelendő füzér kifejezést legalább két részre kell osztani, hogy a rendszer dolgozni tudjon vele.

ILLEGAL DIRECT /Illegális parancs/ - az INPUT utasitás csak programon belül használható, parancsmódban nem.

ILLEGAL QUANTITY /Illegális mennyiség/ - utasitás vagy függvény argumentumaként használt szám kivül esik a megengedett tartományon.

LOAD /Betöltés/ - a szalagon lévő programmal probléma van.

NEXT WITHOUT FOR /FOR nélküli NEXT/ - egymásba hibásan skatulyázott ciklusok álltak vagy az NEXT utasitásban levő és a FOR utasitásban megadott változónév nem egyezése által okozott hiba.

NOT INPUT FILE /Nem bemeneti adatállomány/ - az INPUT vagy GET utasitásokkal akartunk adatokat beolvasni olyan állományból, amely csak kimenetiként lett specifikálva.

NOT OUTPUT FILE /Nem kimeneti állomány/ - a PRINT utasitással akartunk kivinni olyan adatállományra, amely csak bemenetiként lett specifikálva.

OUT OF DATA /Adatból kifutott/ - READ utasitás került végrehajtásra, de a DATA utasitásban nem volt már adat.

OUT OF MEMORY /Betelt a tár/ - nem áll több tárhely rendelkezésre a program vagy a változók részére. Ez adódhat abból is, hogy tul sok FOR ciklus van egymásba skatulyázva vagy valójában tul sok a használt GOSUB.

OVERFLOW /Tulcsordulás/ - a számitás eredménye nagyobb, mint a megengedett legnagyobb szám, amely 1.70141884E+38.

REDIM'D ARRAY / Ujradimenzionált tömb/ - egy tömböt csak egyszer lehet dimenzionálni. Ha egy tömbváltozót a tömb dimenzionálása előtt használunk, egy automatikus DIM müvelet lesz végrehajtva - a tömb elemszáma ll lesz - és a rákövetkező DIM utasitások ezt a hibát hozzák létre.

REDO FROM START /Bemeneti adat hiba/ - INPUT utasitásban karakter adat lett begépelve, amikor számadatot várt. A program magától folytatódik, ha a helyes bemenő értéket ujra begépeli.

RETURN WITHOUT GOSUB /GOSUB nélküli RETURN/ - RETURN utasitást talált a program, és nem volt előtte GOSUB utasitás kiadva.

STRING TOO LONG /Tul hosszu füzér/- a füzér 255 karakternél hosszabb.

?SYNTAX ERROR /Szintaktikus hiba/ - a Commodore 64 által az utasitás felismerhetetlen. Hiányzó vagy felesleges zárójelek, elirt kulcsszavak, stb.

TYPE MISMATCH /Tipus meg nem felelés/ - e hiba abból adódik, ha füzér helyett számot használtunk, és viszont.

UNDEF'D STATEMENT / Definiálatlan függvény/ - egy felhasználó által definiált függvényre történt hivatkozás, de sosem lett a DEF FN utasitással definiálva.

UNDEF'D STATEMENT /Definiálatlan utasitás/ - a GOTO vagy GOSUB vagy RUN egy nem létező sorszámra utal.

VERIFY /Összehasonlitás/ - a szalagon vagy lemezen levő program a tárban levővel nem egyezik.

HANGJEGY ÉRTÉKEK

Ez a függelék a hangjegyek teljes listáját tartalmazza, valamint azokat az értékeket, amelyeket a hang integrált áramkör magasfrekvencia és alacsony frekvencia regisztereibe kell beirni POKE utasitással, hogy megkapjuk a jelzett hangot.

Sorszám	Hangjegy	Al.frekv.	Mag.frefv.
0	C-0	1	18
1	C#-0	1	35
2	D-0	1	52
3	D#-0	1	70
4	E-0	1	90
5	F-0	1	110
6	F#-0	1	132
7	G-0	1	155
8	G#-0	1	179
9	A-0	1	205
10	A#-0	1	233
11	B-0	2	6
12	C-1	2	37
13	C#-1	2	69
14	D-1	2	104
15	D#-1	2	140
16	E-1	2	179
17	F-1	2	220
18	F#-1	3	8
19	G-1	3	54
20	G#-1	3	103
21	A-1	3	155
22	A#-1	3	210
23	B-1	4	12
24	C-2	4	73
25	C#-2	4	139
26	D-2	4	208
27	D#-2	5	25
28	E-2	5	103
29	F-2	5	185
30	F#-2	6	16
31	G-2	6	108
32	G#-2	6	206

Sorszám	Hangjegy	Al.frekv.	Mag.frekv.
33	A-2	7	53
34	A#-2	7	163
35	B-2	8	23
36	C-3	8	147
37	C#-3	9	21
38	D-3	9	159
39	D#-3	10	60
40	E-3	10	205
41	F-3	11	114
42	F#-3	12	32
43	G-3	12	216
44	G#-3	13	156
45	A-3	14	107
46	A#-3	15	70
47	B-3	16	47
48	C-4	17	37
49	C#-4	18	42
50	D-4	19	63
51	D#-4	20	100
52	E-4	21	154
53	F-4	22	227
54	F#-4	24	63
55	G-4	25	177
56	G#-4	27	56
57	A-4	28	214
58	A#-4	30	141
59	B-4	32	94
60	C-5	34	75
61	C#-5	36	85
62	D-5	38	126
63	D#-5	40	200
64	E-5	43	52
65	F-5	45	198
66	F#-5	48	127
67	G-5	51	97
68	G#-5	54	111
69	A-5	57	172
70	A#-5	61	126
71	B-5	64	188
72	C-6	68	149
73	C#-6	72	169
74	D-6	76	252
75	D#-6	81	161
76	E-6	86	105
77	F6	91	140
78	F#-6	96	254
79	G-6	102	194
80	G#-6	108	223

Sorszám	Hangjegy	Al.frekv.	Mag.frekv.
81	A-6	115	88
82	A#-6	122	52
83	B-6	129	120
84	C-7	137	43
85	C#-7	145	83
86	D-7	153	247
87	D#-7	163	31
88	E-7	172	210
89	F-7	183	25
90	F#-7	193	252
91	G-7	205	133
92	G#-7	217	189
93	A-7	230	176
94	A#-7	244	103

IRODALOM

1. Donald Alcock: Ismerd meg a BASIC nyelvet!, Műszaki Könyvkiadó, Budapest, 1983.

2. Bodor Tibor-Gerő Péter: A BASIC programozás technikája. SZÁMALK, Budapest, 1983.

3. Dr. Kocsis András: Programozás BASIC nyelven I-II-III. SZÁMALK, Budapest, 1983.

4. COMMODORE-The Microcomputer Magazine, havonta jelenik meg.

5. POWER PLAY-Computing Magazine, negyedévenként jelenik meg.

SZELLEM REGISZTER TÉRKÉP

Báziscím=53248$_{dec}$=D000$_{hex}$

Register # Dec	Hex	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	S0X7	S0X6	S0X5	S0X4	S0X3	S0X2	S0X1	S0X0	0 Szellem, X
1	1	S0Y7							S0Y0	0 Szellem, Y
2	2	S1X7							S1X0	1 Szellem, X
3	3	S1Y7							S1Y0	1 Szellem, Y
4	4	S2X7							S2X0	2 Szellem, X
5	5	S2Y7							S2Y0	2 Szellem, Y
6	6	S3X7							S3X0	3 Szellem, X
7	7	S3Y7							S3Y0	3 Szellem, Y
8	8	S4X7							S4X0	4 Szellem, X
9	9	S4Y7							S4Y0	4 Szellem, Y
10	A	S5X7							S5X0	5 Szellem, X
11	B	S5Y7							S5Y0	5 Szellem, Y
12	C	S6X7							S6X0	6 Szellem, X
13	D	S6Y7							S6Y0	6 Szellem, Y
14	E	S7X7							S7X0	7 Szellem, X
15	F	S7Y7							S7Y0	7 Szellem, Y
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	X koordináta MSB-je
17	11	RC8	EC5	BSM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Raszter
19	13	LPX7							LPX0	Fényceruza, X
20	14	LPY7							LPY0	Fényceruza, Y
21	15	SE7							SE0	Szellem be/ki kapcsolás
22	16	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	
23	17	SEXY7							SEXY0	Szellemkiterjesztés, Y

Register # Dec	Hex	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
24	18	VS13	VS12	VS11	CB13	CB12	CB11	CB10	N.C.	Képernyő karakter tár
25	19	IRQ	N.C.	N.C.	N.C.	LPIRQ	ISSC	ISBC	RIRQ	Megszakítás kérés
26	1A	N.C.	N.C.	N.C.	N.C.	MLPI	MISSC	MISBC	MRIRQ	Megszakítás kérés maszkok
27	1B	BSP7							BSP0	Háttér-Szellem prioritás
28	1C	SCM7							SCM0	Multikolor szellem kiválasztás
29	1D	SEXX7							SEXX0	Szellem kiterjesztés X
30	1E	SSC7							SSC0	Szellem-szellem ütközés
31	1F	SBC7							SBC0	Szellem-háttér ütközés

SZÍNINFORMÁCIÓK

32	20	EXT 1				Keret szin
33	21	BKGD0				Háttér 1
34	22	BKGD1				2
35	23	BKGD2				3
36	24	BKGD3				4
37	25	SMC 0				Szellem multikolor 0
38	26	SMC 1				1
39	27	S0COL				0 szellem
40	28	S1COL				1
41	29	S2COL				2
42	2A	S3COL				3
43	2B	S4COL				4
44	2C	S5COL				5
45	2D	S6COL				6
46	2E	S7COL				7

COMMODORE 64 HANG VEZÉRLŐ BEÁLLITÁSOK

Ez a táblázat megadja azokat a kulcsszámokat, amelyek az Ön hang programjához szükségesek, annak megfelelően, hogy a 3 szólam melyikét kívánja használni. A hangvezérlés beállitásához vagy szabályozásához BASIC programjában irjon egy POKE utasitást, majd a második oszlopból vett számot, utána vesszőt és a táblázatból vett számot, úgymint: POKE 54276,17 /kiválaszt az 1. szólamra egy háromszög hullámalakot/.

Emlékezzen arra, hogy elősz̈r a hangerőt kell beállitani mielőtt hangot generálhatna. Mindhárom szólamra a POKE 54296-t egy Ø-tól 15-ig terjedő szám beállitása követi.

Minden egyes hangot 2 különböző POKE határoz meg, pl. POKE 54273,34: POKE 54272,75 adja a következő táblázatbeli minta skála alsó C hangját.

A táblázatban megadott számok nem kötelezőek. Ha az előbb kapott C hang "hamisan" szól, 34 helyett próbálja ki a 35-t. A bemutatottnál hosszabb tartáshoz vagy felfutáshoz adjon össze két vagy több tartás értéket. /Pl. POKE 54277,96 két felfutás értéket ad össze egy nagyobb felfutás elérésére, de pl. POKE 54277,2Ø egy alacsony felfutás értéket /16/ és egy közepes lecsengést /4/ ad.

TÁBLÁZAT

(a következő oldalon)

Funkció	Szólam	Tárhely	Tartalom													
Hangerő	1,2,3	54296	0 (csend) -től 15 (hangos) -ig													
Hangjegy leját.			C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
Magas frekvencia	1 2 3	54273 54280 54287	34	36	38	40	43	45	48	51	54	57	61	64	68	72
Alacsony frekvencia	1 2 3	54272 54279 54286	75	85	126	200	52	198	127	97	111	172	126	188	149	169
			Háromszög			Fűrészfog			Impulzus			Zaj				
Hullámalak	1 2 3	54276 54283 54290	17			33			65			129				
Impulzus alak																
Felső rész	1 2 3	54275 54282 54289							0-15							
Alsó rész	1 2 3	54274 54281 54288							0-255							
			F4	F3	F2	F1	L4	L3	L2	L1						
Felfutás-lecsengés	1 2 3	54277 54284 54291	128	64	32	16	8	4	2	1						
			T4	T3	T2	T1	E4	E3	E2	E1						
Tartás-elengedés	1 2 3	54278 54285 54292	128	64	32	16	8	4	2	1						

Próbálja ezekkel a beállitásokkal szimulálni a különböző hangszereket:

Hangszer	Hullám-alak tipusa	Felfutás/ lecsengés	Kitartás/ elengedés	Impulzus kitöltési tényező
Zongora	Impulzus	9	Ø	Magas-O Alacsony 255
Furulya	Háromszog	96	Ø	Nem alkalmazható
Csembaló	Fürészfog	9	Ø	"
Xilofon	Háromszög	9	Ø	"
Orgona	Háromszög	Ø	24Ø	"
Harmonium	Háromszög	Ø	24Ø	"
Harmonika	Háromszög	1Ø2	Ø	
Trombita	Fürészfog	96	Ø	"

A használatos szavak értelmezése

FLTE - felfutás/lecsengés/kitartás/elengedés

felfutás - a legnagyobb hangerő értékre való felfutás sebessége

lecsengés - a legnagyobb hangerő értékről való csökkenés sebessége

tartás - egy bizonyos hangerőn meghosszabbitja a hangot.

elengedés - a tartott hangerő szintről a nulla hangerőre való csökkenés sebessége

KÖSZÖNÖM

hullámalak - a hanghullám alakja

impulzus kitöltési tényező - az impulzus hullámalak hangminőségé

Megjegyzés:

A felfutás/lecsengés és tartás/elengedés beállitást mindig a hullámalak beállitása előtt kell elvégezni.

commodore 64